

EVENT: Start with the library "c-predefined3".

THEOREM: arrays-have-non-zero-lengths
 $(\text{array-identifierp}(x, \text{alist}) \wedge \text{mg-alistp}(\text{alist}))$
 $\rightarrow (\text{array-length}(\text{cadr}(\text{assoc}(x, \text{alist}))) \neq 0)$

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;                                EXACT-TIME LEMMA MG-INDEX-ARRAY
;;
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

EVENT: Enable map-down-values-preserves-length.

THEOREM: lessp-plus-transitive
 $((x + (w - 1)) < z) \wedge (y < w) \rightarrow ((x + y) < z) = \mathbf{t})$

EVENT: Disable lessp-plus-transitive.

THEOREM: idifference-lessp2
 $((x \in \mathbf{N}) \wedge (x < y))$
 $\rightarrow (((\text{idifference}(y, x) = 0) = \mathbf{f}) \wedge (\text{idifference}(y, x) \in \mathbf{N}))$

THEOREM: zerop-integerp-trichotomy
 $(\text{integerp}(x) \wedge (x \notin \mathbf{N})) \rightarrow \text{negativep}(x)$

EVENT: Disable zerop-integerp-trichotomy.

THEOREM: p-object-type-int
 $(\text{p-objectp-type}(' \text{int}, \text{tag}(' \text{int}, x), \text{state}))$
 $= \text{small-integerp}(x, \text{p-word-size}(\text{state}))$
 $\wedge (\text{p-objectp-type}(' \text{int}, \text{list}(' \text{int}, x), \text{state}))$
 $= \text{small-integerp}(x, \text{p-word-size}(\text{state}))$

THEOREM: simple-identifierp-options
 $(\text{int-identifierp}(x, \text{alist}))$
 $\vee \text{boolean-identifierp}(x, \text{alist})$
 $\vee \text{character-identifierp}(x, \text{alist})$
 $\rightarrow \text{simple-identifierp}(x, \text{alist})$

THEOREM: small-integerp-difference
 $(\text{small-integerp}(x, n))$

$\wedge$ `small-integerp` (y , n)
 $\wedge$ ($x \not\approx 0$)
 $\wedge$ ($\neg$ `negativep` (y))
 $\rightarrow$ `small-integerp` (`idifference` (x , y), n)

THEOREM: limits-for-small-integerp
 $((x < \text{MAXINT}) \wedge (x \not\approx 0)) \rightarrow \text{small-integerp}(x, 32)$

THEOREM: simple-typed-identifier-simple-identifierp
`simple-typed-identifierp` (x , $type$, $alist$) $\rightarrow$ `simple-identifierp` (x , $alist$)

THEOREM: mg-var-ok-array-index-ok3
`(mg-vars-list-ok-in-p-state` (mg -vars, $bindings$, $temp$ -stk)
 $\wedge$ `mg-alistp` (mg -vars)
 $\wedge$ `array-identifierp` (x , mg -vars))
 $\rightarrow$ $((\text{untag}(\text{value}(x, bindings))$
 $\quad + (\text{array-length}(\text{cadr}(\text{assoc}(x, mg\text{-vars}))) - 1))$
 $\quad < \text{length}(temp\text{-stk}))$
 $= \text{t})$

THEOREM: idifference-lessp
 $((\neg \text{negativep}(y)) \wedge (\text{idifference}(x, y) \not\approx 0)) \rightarrow ((y < x) = \text{t})$

THEOREM: nat-p-objectp-reduction
`p-objectp-type` (`'nat`, `tag` (`'nat`, x), $state$)
 $=$ `small-naturalp` (x , `p-word-size` ($state$))

THEOREM: array-index-small-naturalp
 $((temp\text{-stk-size} < \text{MG-MAX-TEMP-STK-SIZE})$
 $\wedge ((a + (\text{array-size} - 1)) < temp\text{-stk-size})$
 $\wedge (index < \text{array-size}))$
 $\rightarrow$ `small-naturalp` ($a + index$, 32)

THEOREM: mg-index-array-args-have-simple-mg-type-refps
 $((\text{car}(stmt) = \text{'predefined-proc-call-mg})$
 $\wedge (\text{call-name}(stmt) = \text{'mg-index-array})$
 $\wedge \text{ok-mg-statement}(stmt, r\text{-cond-list}, name\text{-alist}, proc\text{-list})$
 $\wedge \text{ok-mg-statep}(mg\text{-state}, r\text{-cond-list})$
 $\wedge \text{signatures-match}(mg\text{-alist}(mg\text{-state}), name\text{-alist}))$
 $\rightarrow$ $(\text{simple-typed-identifierp}(\text{car}(\text{call-actuals}(stmt)),$
 $\quad \text{array-elemtype}(\text{cadr}(\text{assoc}(\text{cadr}(\text{call-actuals}(stmt)),$
 $\quad \quad \text{mg-alist}(mg\text{-state}))),$
 $\quad \text{mg-alist}(mg\text{-state}))$
 $\wedge \text{array-identifierp}(\text{cadr}(\text{call-actuals}(stmt)), \text{mg-alist}(mg\text{-state}))$
 $\wedge \text{int-identifierp}(\text{caddr}(\text{call-actuals}(stmt)), \text{mg-alist}(mg\text{-state}))$

$$\begin{aligned} & \wedge \quad (\text{caddr}(\text{call-actuals}(stmt))) \\ & = \quad \text{array-length}(\text{cadr}(\text{assoc}(\text{cadr}(\text{call-actuals}(stmt)), \\ & \quad \text{mg-alist}(mg\text{-state})))))) \end{aligned}$$

THEOREM: `mg-index-array-args-definedp`

```
((car (stmt) = 'predefined-proc-call-mg)
  ∧ (call-name (stmt) = 'mg-index-array)
  ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
  ∧ ok-mg-statep (mg-state, r-cond-list)
  ∧ signatures-match (mg-alist (mg-state), name-alist))
→ (definedp (car (call-actuals (stmt))), mg-alist (mg-state))
    ∧ definedp (cadr (call-actuals (stmt))), mg-alist (mg-state))
    ∧ definedp (caddr (call-actuals (stmt))), mg-alist (mg-state)))
```

THEOREM: mg-index-array-arg4-small-integerp

```
((car (stmt) = 'predefined-proc-call-mg)
  ∧ (call-name (stmt) = 'mg-index-array)
  ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
  ∧ ok-mg-statep (mg-state, r-cond-list)
  ∧ signatures-match (mg-alist (mg-state), name-alist))
→ small-integerp (caddr (call-actuals (stmt)), 32)
```

THEOREM: not-zerop-mg-index-array-arg4

$$\begin{aligned} & ((\text{car}(stmt) = \text{'predefined-proc-call-mg'}) \\ & \wedge (\text{call-name}(stmt) = \text{'mg-index-array'}) \\ & \wedge \text{ok-mg-statement}(stmt, r\text{-cond-list}, name\text{-alist}, proc\text{-list}) \\ & \wedge \text{ok-mg-statep}(mg\text{-state}, r\text{-cond-list}) \\ & \wedge \text{signatures-match}(mg\text{-alist}(mg\text{-state}), name\text{-alist})) \\ \rightarrow & ((\text{caddr}(\text{call-actuals}(stmt)) \in \mathbf{N}) \\ & \wedge (\text{caddr}(\text{call-actuals}(stmt)) \neq 0)) \end{aligned}$$

THEOREM: mg-index-array-steps-1-4

$$\begin{aligned} & ((n \neq 0) \\ & \wedge (\neg \text{resources-inadequatep} (stmt, \\ & \qquad \qquad \qquad \text{proc-list}, \\ & \qquad \qquad \qquad \text{list (length (temp-stk), \\ & \qquad \qquad \qquad \text{p-ctrl-stk-size (ctrl-stk)}))) \\ & \wedge (\text{car} (stmt) = \text{'predefined-proc-call-mg}) \\ & \wedge (\text{call-name} (stmt) = \text{'mg-index-array}) \\ & \wedge \text{ok-mg-statement} (stmt, r\text{-cond-list}, name\text{-alist}, proc\text{-list}) \\ & \wedge \text{ok-mg-def-plistp} (proc\text{-list}) \\ & \wedge \text{ok-mg-statep} (mg\text{-state}, r\text{-cond-list}) \\ & \wedge (\text{code} (\text{translate-def-body} (\text{assoc} (subr, proc\text{-list}), proc\text{-list})) \\ & \quad = \text{append} (\text{code} (\text{translate} (cinfo, t\text{-cond-list}, stmt, proc\text{-list})), \\ & \quad \quad \text{code2}))) \end{aligned}$$

$$\begin{aligned}
& \wedge \text{ user-defined-procp } (subr, proc\text{-}list) \\
& \wedge \text{ listp } (ctrl\text{-}stk) \\
& \wedge \text{ all-cars-unique } (mg\text{-}alist (mg\text{-}state)) \\
& \wedge \text{ signatures-match } (mg\text{-}alist (mg\text{-}state), name\text{-}alist) \\
& \wedge \text{ mg-vars-list-ok-in-p-state } (mg\text{-}alist (mg\text{-}state), \\
& \quad \text{bindings } (top (ctrl\text{-}stk)), \\
& \quad \text{temp-stk}) \\
& \wedge \text{ no-p-aliasing } (bindings (top (ctrl\text{-}stk)), mg\text{-}alist (mg\text{-}state)) \\
& \wedge \text{ normal } (mg\text{-}state)) \\
\rightarrow & \text{ (p-step (p-step (p-step (p-step (map-down } (mg\text{-}state, \\
& \quad \text{proc-list,} \\
& \quad \text{ctrl-stk,} \\
& \quad \text{temp-stk,} \\
& \quad \text{tag ('pc,} \\
& \quad \quad \text{cons (subr,} \\
& \quad \quad \quad \text{length (code (cinfo))))),} \\
& \quad \quad \text{t-cond-list)))))) \\
= & \text{ p-state (tag ('pc, cons (subr, length (code (cinfo)) + 4)),} \\
& \quad \text{ctrl-stk,} \\
& \quad \text{push (tag ('int, caddr (call-actuals (stmt)),} \\
& \quad \quad \text{push (value (caddr (call-actuals (stmt)),} \\
& \quad \quad \quad \text{bindings (top (ctrl-stk))),} \\
& \quad \quad \text{push (value (cadr (call-actuals (stmt)),} \\
& \quad \quad \quad \text{bindings (top (ctrl-stk))),} \\
& \quad \quad \text{push (value (car (call-actuals (stmt)),} \\
& \quad \quad \quad \text{bindings (top (ctrl-stk))),} \\
& \quad \quad \text{map-down-values (mg-alist (mg-state),} \\
& \quad \quad \quad \text{bindings (top (ctrl-stk)),} \\
& \quad \quad \quad \text{temp-stk))))),} \\
& \quad \text{translate-proc-list (proc-list),} \\
& \quad \text{list (list ('c-c,} \\
& \quad \quad \text{mg-cond-to-p-nat (cc (mg-state), t-cond-list))),} \\
& \quad \text{MG-MAX-CTRL-STK-SIZE,} \\
& \quad \text{MG-MAX-TEMP-STK-SIZE,} \\
& \quad \text{MG-WORD-SIZE,} \\
& \quad \text{'run}))
\end{aligned}$$

THEOREM: mg-index-array-step-5

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge \quad (\neg \text{ resources-inadequatep } (stmt, \\
& \quad \quad \text{proc-list,} \\
& \quad \quad \text{list (length (temp-stk),} \\
& \quad \quad \quad \text{p-ctrl-stk-size (ctrl-stk)))) \\
& \wedge \quad (\text{car } (stmt) = \text{'predefined-proc-call-mg})
\end{aligned}$$

```

^ (call-name (stmt) = 'mg-index-array)
^ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
^ ok-mg-def-plistp (proc-list)
^ ok-mg-statep (mg-state, r-cond-list)
^ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
        code2))
^ user-defined-procp (subr, proc-list)
^ listp (ctrl-stk)
^ all-cars-unique (mg-alist (mg-state))
^ signatures-match (mg-alist (mg-state), name-alist)
^ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
    bindings (top (ctrl-stk)),
    temp-stk)
^ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
^ normal (mg-state))
→ (p-step (p-state (tag ('pc, cons (subr, length (code (cinfo)) + 4)),
    ctrl-stk,
    push (tag ('int, caddr (call-actuals (stmt))),
        push (value (caddr (call-actuals (stmt)),
            bindings (top (ctrl-stk))),
            push (value (cadr (call-actuals (stmt)),
                bindings (top (ctrl-stk))),
                push (value (car (call-actuals (stmt)),
                    bindings (top (ctrl-stk))),
                    map-down-values (mg-alist (mg-state),
                        bindings (top (ctrl-stk)),
                        temp-stk))))),
        translate-proc-list (proc-list),
        list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
        MG-MAX-CTRL-STK-SIZE,
        MG-MAX-TEMP-STK-SIZE,
        MG-WORD-SIZE,
        'run))
    = p-state (tag ('pc, ' (mg-index-array . 0)),
        push (p-frame (cons (cons ('ans,
            value (car (call-actuals (stmt)),
                bindings (top (ctrl-stk)))),
            cons (cons ('a,
                value (cadr (call-actuals (stmt)),
                    bindings (top (ctrl-stk)))),
                cons (cons ('i,
                    value (caddr (call-actuals (stmt))),

```

```

bindings (top (ctrl-stk))),
cons (cons ('array-size,
tag ('int,
caddr (call-actuals (stmt))),
'((temp-i nat 0)))))
tag ('pc,
cons (subr, length (code (cinfo)
+ 5))),
ctrl-stk),
map-down-values (mg-alist (mg-state),
bindings (top (ctrl-stk)),
temp-stk),
translate-proc-list (proc-list),
list (list ('c-c,
mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-index-array-steps-6-8

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
proc-list,
list (length (temp-stk),
p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-index-array)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp (proc-list)
 ∧ ok-mg-statep (mg-state, r-cond-list)
 ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
 = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
code2))
 ∧ user-defined-procp (subr, proc-list)
 ∧ listp (ctrl-stk)
 ∧ all-cars-unique (mg-alist (mg-state))
 ∧ signatures-match (mg-alist (mg-state), name-alist)
 ∧ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
bindings (top (ctrl-stk)),
temp-stk)
 ∧ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 ∧ normal (mg-state))
 → (p-step (p-step (p-step (p-state (tag ('pc, ' (mg-index-array . 0))),

```

```

push (p-frame (cons (cons ('ans,
                           value (car (call-actuals (stmt)),
                           bindings (top (ctrl-stk)))))
                  cons (cons ('a,
                              value (cadr (call-actuals (stmt)),
                              bindings (top (ctrl-stk)))))
                  cons (cons ('i,
                              value (caddr (call-actuals (stmt)),
                              bindings (top (ctrl-stk)))))
                  cons (cons ('array-size,
                              tag ('int,
                                   caddr (call-actuals (stmt)
                                   '((temp-i
                                     nat
                                     0)))))),
                  tag ('pc,
                      cons (subr,
                          length (code (cinfo)
                          + 5))),
                      ctrl-stk),
map-down-values (mg-alist (mg-state),
                  bindings (top (ctrl-stk)),
                  temp-stk),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-state),
                                t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))))
= p-state (tag ('pc, '(mg-index-array . 3)),
           push (p-frame (list (cons ('ans,
                                      value (car (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))))
                              cons ('a,
                                      value (cadr (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))))
                              cons ('i,
                                      value (caddr (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))))
                              cons ('array-size,
                                      tag ('int,
                                           caddr (call-actuals (stmt)
                                           '((temp-i
                                             nat
                                             0)))))),
                              ctrl-stk),
            map-down-values (mg-alist (mg-state),
                              bindings (top (ctrl-stk)),
                              temp-stk),
            translate-proc-list (proc-list),
            list (list ('c-c,
                        mg-cond-to-p-nat (cc (mg-state),
                                              t-cond-list))),
            MG-MAX-CTRL-STK-SIZE,
            MG-MAX-TEMP-STK-SIZE,
            MG-WORD-SIZE,
            'run))))

```

```

cons('temp-i,
     mg-to-p-simple-literal(caddr(assoc(caddr(call-actuals(stmt)),
                                           mg-alist(mg-state)))))),
tag('pc,
    cons(subr, length(code(cinfo)
                          + 5))),
ctrl-stk),
push(mg-to-p-simple-literal(caddr(assoc(caddr(call-actuals(stmt)),
                                           mg-alist(mg-state))))),
     map-down-values(mg-alist(mg-state),
                      bindings(top(ctrl-stk),
                                temp-stk))),
translate-proc-list(proc-list),
list(list('c-c,
          mg-cond-to-p-nat(cc(mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-index-array-steps-9-12-neg-index

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep(stmt,
                             proc-list,
                             list(length(temp-stk),
                                     p-ctrl-stk-size(ctrl-stk))))
 ∧ (car(stmt) = 'predefined-proc-call-mg)
 ∧ (call-name(stmt) = 'mg-index-array)
 ∧ ok-mg-statement(stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp(proc-list)
 ∧ ok-mg-statep(mg-state, r-cond-list)
 ∧ (code(translate-def-body(assoc(subr, proc-list), proc-list))
    = append(code(translate(cinfo, t-cond-list, stmt, proc-list)),
              code2))
 ∧ user-defined-procp(subr, proc-list)
 ∧ listp(ctrl-stk)
 ∧ all-cars-unique(mg-alist(mg-state))
 ∧ signatures-match(mg-alist(mg-state), name-alist)
 ∧ mg-vars-list-ok-in-p-state(mg-alist(mg-state),
                               bindings(top(ctrl-stk),
                                         temp-stk))
 ∧ no-p-aliasing(bindings(top(ctrl-stk), mg-alist(mg-state))
 ∧ normal(mg-state)
 ∧ negativep(untag(mg-to-p-simple-literal(caddr(assoc(caddr(call-actuals(stmt)),

```

```

→ (p-step (p-step (p-step (p-step (p-state (tag ('pc,
                                             '(mg-index-array . 3)),
                                             push (p-frame (list (cons ('ans,
                                                                      value (car (call-actuals (stmt)),
                                                                      bindings (top (ctrl-stk)))),
                                                                      cons ('a,
                                                                      value (cadr (call-actuals (stmt)),
                                                                      bindings (top (ctrl-stk)))),
                                                                      cons ('i,
                                                                      value (caddr (call-actuals (stmt)),
                                                                      bindings (top (ctrl-stk)))),
                                                                      cons ('array-size,
                                                                      tag ('int,
                                                                      caddr (call-actuals (stmt)))),
                                                                      cons ('temp-i,
                                                                      mg-to-p-simple-literal (caddr (assoc (caddr
                                                                                                     mg-
                                                                                                     mg-
                                                                                                     tag ('pc,
                                                                                                     cons (subr,
                                                                                                     length (code (cinfo))
                                                                                                     + 5))),
                                                                                                     ctrl-stk),
                                                                                                     push (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)
                                                                                                     mg-
                                                                                                     mg-
                                                                                                     mg-
                                                                                                     map-down-values (mg-alist (mg-state),
                                                                                                     bindings (top (ctrl-stk)),
                                                                                                     temp-stk)),
                                                                                                     translate-proc-list (proc-list),
                                                                                                     list (list ('c-c,
                                                                                                     mg-cond-to-p-nat (cc (mg-state),
                                                                                                     t-cond-list))),
                                                                                                     MG-MAX-CTRL-STK-SIZE,
                                                                                                     MG-MAX-TEMP-STK-SIZE,
                                                                                                     MG-WORD-SIZE,
                                                                                                     'run))))))
                                             ctrl-stk,
                                             map-down-values (mg-alist (mg-state),
                                             bindings (top (ctrl-stk)),
                                             temp-stk),
                                             translate-proc-list (proc-list),
                                             list (list ('c-c, 'nat 1)),
                                             MG-MAX-CTRL-STK-SIZE,
                                             'run))))))
= p-state (tag ('pc, cons (subr, length (code (cinfo)) + 5)),
          ctrl-stk,
          map-down-values (mg-alist (mg-state),
                          bindings (top (ctrl-stk)),
                          temp-stk),
          translate-proc-list (proc-list),
          list (list ('c-c, 'nat 1)),
          MG-MAX-CTRL-STK-SIZE,
          'run))

```

MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: mg-index-array-steps-9-11-no-error

(($n \neq 0$)
 $\wedge$ ($\neg$ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
 $\wedge$ (car (*stmt*) = 'predefined-proc-call-mg)
 $\wedge$ (call-name (*stmt*) = 'mg-index-array)
 $\wedge$ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
 $\wedge$ ok-mg-def-plistp (*proc-list*)
 $\wedge$ ok-mg-statep (*mg-state*, *r-cond-list*)
 $\wedge$ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
 $\wedge$ user-defined-procp (*subr*, *proc-list*)
 $\wedge$ listp (*ctrl-stk*)
 $\wedge$ all-cars-unique (mg-alist (*mg-state*))
 $\wedge$ signatures-match (mg-alist (*mg-state*), *name-alist*)
 $\wedge$ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
 $\wedge$ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
 $\wedge$ normal (*mg-state*)
 $\wedge$ ($\neg$ negativep (untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (*stmt*)),
mg-alist (*mg-state*))))))))
 $\rightarrow$ (p-step (p-step (p-step (p-state (tag ('pc, 'mg-index-array . 3)),
push (p-frame (list (cons ('ans,
value (car (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*))),
cons ('a,
value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*))),
cons ('i,
value (caddr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*))),
cons ('array-size,
tag ('int,
caddr (call-actuals (*stmt*))),
cons ('temp-i,
mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (*stmt*)),
mg-alist (*mg-state*))))))))))))))

```

tag('pc,
    cons(subr,
        length(code(cinfo))
        + 5)),
ctrl-stk),
push(mg-to-p-simple-literal(caddr(assoc(caddr(call-actuals(stmt)),
                                         mg-alist(mg-state))))),
    map-down-values(mg-alist(mg-state),
        bindings(top(ctrl-stk)),
        temp-stk)),
translate-proc-list(proc-list),
list(list('c-c,
    mg-cond-to-p-nat(cc(mg-state),
        t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))))
= p-state(tag('pc, '(mg-index-array . 6)),
    push(p-frame(list(cons('ans,
        value(car(call-actuals(stmt)),
            bindings(top(ctrl-stk))))),
    cons('a,
        value(cadr(call-actuals(stmt)),
            bindings(top(ctrl-stk))))),
    cons('i,
        value(caddr(call-actuals(stmt)),
            bindings(top(ctrl-stk))))),
    cons('array-size,
        tag('int,
            caddr(call-actuals(stmt)))),
    cons('temp-i,
        mg-to-p-simple-literal(caddr(assoc(caddr(call-actuals(stmt)),
                                         mg-alist(mg-state)))))),
tag('pc,
    cons(subr, length(code(cinfo))
        + 5)),
ctrl-stk),
push(mg-to-p-simple-literal(caddr(assoc(caddr(call-actuals(stmt)),
                                         mg-alist(mg-state))))),
push(tag('int, caddr(call-actuals(stmt))),
    map-down-values(mg-alist(mg-state),
        bindings(top(ctrl-stk))),

```

```

                                temp-stk))),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

```

;; This is the (sub-int) instruction
;; The proof of this could be cleaned up substantially

```

THEOREM: mg-index-array-step-12-no-error

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
                             proc-list,
                             list (length (temp-stk),
                                     p-ctrl-stk-size (ctrl-stk)))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-index-array)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp (proc-list)
 ∧ ok-mg-statep (mg-state, r-cond-list)
 ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
      = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                code2))
 ∧ user-defined-procp (subr, proc-list)
 ∧ listp (ctrl-stk)
 ∧ all-cars-unique (mg-alist (mg-state))
 ∧ signatures-match (mg-alist (mg-state), name-alist)
 ∧ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                               bindings (top (ctrl-stk)),
                               temp-stk)
 ∧ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 ∧ normal (mg-state)
 ∧ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                             mg-alist (mg-state))))))))
 → (p-step (p-state (tag ('pc, '(mg-index-array . 6)),
                    push (p-frame (list (cons ('ans,
                                              value (car (call-actuals (stmt)),
                                              bindings (top (ctrl-stk)))),
                                              cons ('a,
                                              value (cadr (call-actuals (stmt))),

```

```

bindings (top (ctrl-stk))),
cons ('i,
      value (caddr (call-actuals (stmt)),
              bindings (top (ctrl-stk))),
cons ('array-size,
      tag ('int,
            caddr (call-actuals (stmt))),
cons ('temp-i,
      mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                              mg-alist (mg-state)))))),
tag ('pc,
     cons (subr, length (code (cinfo))
           + 5))),
ctrl-stk),
push (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                              mg-alist (mg-state))))),
push (tag ('int, caddr (call-actuals (stmt))),
      map-down-values (mg-alist (mg-state),
                          bindings (top (ctrl-stk)),
                          temp-stk))),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
= p-state (tag ('pc, ' (mg-index-array . 7)),
           push (p-frame (list (cons ('ans,
                                      value (car (call-actuals (stmt)),
                                                bindings (top (ctrl-stk))),
cons ('a,
      value (cadr (call-actuals (stmt)),
              bindings (top (ctrl-stk))),
cons ('i,
      value (caddr (call-actuals (stmt)),
              bindings (top (ctrl-stk))),
cons ('array-size,
      tag ('int,
            caddr (call-actuals (stmt))),
cons ('temp-i,
      mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                              mg-alist (mg-state)))))),
tag ('pc,

```

```

cons (subr, length (code (cinfo))
      + 5)),
ctrl-stk),
push (tag ('int,
          idifference (caddr (call-actuals (stmt)),
                      untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt))
                                                                    mg-alist (mg-state))))))),
      map-down-values (mg-alist (mg-state),
                      bindings (top (ctrl-stk)),
                      temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-index-array-step-13-index-error

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
                             proc-list,
                             list (length (temp-stk),
                                     p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-index-array)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp (proc-list)
 ∧ ok-mg-statep (mg-state, r-cond-list)
 ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
              code2))
 ∧ user-defined-procp (subr, proc-list)
 ∧ listp (ctrl-stk)
 ∧ all-cars-unique (mg-alist (mg-state))
 ∧ signatures-match (mg-alist (mg-state), name-alist)
 ∧ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                               bindings (top (ctrl-stk)),
                               temp-stk)
 ∧ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 ∧ normal (mg-state)
 ∧ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                                mg-alist (mg-state)))))))
 ∧ (idifference (caddr (call-actuals (stmt)),

```

```

untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                             mg-alist (mg-state))))))  $\simeq$  0))
→ (p-step (p-state (tag ('pc, '(mg-index-array . 7)),
                     push (p-frame (list (cons ('ans,
                                                value (car (call-actuals (stmt)),
                                                bindings (top (ctrl-stk)))),
                                                cons ('a,
                                                value (cadr (call-actuals (stmt)),
                                                bindings (top (ctrl-stk)))),
                                                cons ('i,
                                                value (caddr (call-actuals (stmt)),
                                                bindings (top (ctrl-stk)))),
                                                cons ('array-size,
                                                tag ('int,
                                                caddr (call-actuals (stmt)))),
                                                cons ('temp-i,
                                                mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                mg-alist (mg-state)))))),
                                                tag ('pc,
                                                cons (subr, length (code (cinfo)
                                                + 5))),
                                                ctrl-stk),
                     push (tag ('int,
                                idifference (caddr (call-actuals (stmt)),
                                untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                mg-alist (mg-state)))))),
                                map-down-values (mg-alist (mg-state),
                                bindings (top (ctrl-stk)),
                                temp-stk)),
                                translate-proc-list (proc-list),
                                list (list ('c-c,
                                mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                                MG-MAX-CTRL-STK-SIZE,
                                MG-MAX-TEMP-STK-SIZE,
                                MG-WORD-SIZE,
                                'run))
    = p-state (tag ('pc, '(mg-index-array . 16)),
              push (p-frame (list (cons ('ans,
                                          value (car (call-actuals (stmt)),
                                          bindings (top (ctrl-stk)))),
                                          cons ('a,
                                          value (cadr (call-actuals (stmt)),
                                          bindings (top (ctrl-stk)))),
                                          cons ('i,

```

```

value (caddr (call-actuals (stmt)),
      bindings (top (ctrl-stk))),
cons ('array-size,
      tag ('int,
            caddr (call-actuals (stmt))),
cons ('temp-i,
      mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                              mg-alist (mg-state))))),
tag ('pc,
     cons (subr, length (code (cinfo)
                               + 5))),
ctrl-stk),
map-down-values (mg-alist (mg-state),
                  bindings (top (ctrl-stk)),
                  temp-stk),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-index-array-steps-14-16-index-error

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
                             proc-list,
                             list (length (temp-stk),
                                       p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-index-array)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp (proc-list)
 ∧ ok-mg-statep (mg-state, r-cond-list)
 ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
              code2))
 ∧ user-defined-procp (subr, proc-list)
 ∧ listp (ctrl-stk)
 ∧ all-cars-unique (mg-alist (mg-state))
 ∧ signatures-match (mg-alist (mg-state), name-alist)
 ∧ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                       bindings (top (ctrl-stk)),
                                       temp-stk)

```

$$\begin{aligned}
& \wedge \text{ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))} \\
& \wedge \text{ normal (mg-state)} \\
& \wedge (\neg \text{ negativep (untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)), \\
& \hspace{15em} \text{mg-alist (mg-state)))))))))) \\
& \wedge (\text{ idifference (caddr (call-actuals (stmt)),} \\
& \hspace{10em} \text{untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),} \\
& \hspace{15em} \text{mg-alist (mg-state))))))} \simeq 0)) \\
\rightarrow & (\text{ p-step (p-step (p-step (p-state (tag ('pc, '(mg-index-array . 16)),} \\
& \hspace{10em} \text{push (p-frame (list (cons ('ans,} \\
& \hspace{15em} \text{value (car (call-actuals (stmt)),} \\
& \hspace{15em} \text{bindings (top (ctrl-stk)))),} \\
& \hspace{10em} \text{cons ('a,} \\
& \hspace{15em} \text{value (cadr (call-actuals (stmt)),} \\
& \hspace{15em} \text{bindings (top (ctrl-stk)))),} \\
& \hspace{10em} \text{cons ('i,} \\
& \hspace{15em} \text{value (caddr (call-actuals (stmt)),} \\
& \hspace{15em} \text{bindings (top (ctrl-stk)))),} \\
& \hspace{10em} \text{cons ('array-size,} \\
& \hspace{15em} \text{tag ('int,} \\
& \hspace{15em} \text{caddr (call-actuals (stmt))))}, \\
& \hspace{10em} \text{cons ('temp-i,} \\
& \hspace{15em} \text{mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),} \\
& \hspace{15em} \text{mg-alist (mg-state))))} \\
& \hspace{10em} \text{tag ('pc,} \\
& \hspace{10em} \text{cons (subr,} \\
& \hspace{10em} \text{length (code (cinfo))} \\
& \hspace{10em} \text{+ 5))),} \\
& \hspace{10em} \text{ctrl-stk}), \\
& \text{ map-down-values (mg-alist (mg-state),} \\
& \hspace{10em} \text{bindings (top (ctrl-stk)),} \\
& \hspace{10em} \text{temp-stk}), \\
& \text{ translate-proc-list (proc-list),} \\
& \text{ list (list ('c-c,} \\
& \hspace{10em} \text{mg-cond-to-p-nat (cc (mg-state),} \\
& \hspace{10em} \text{t-cond-list))),} \\
& \text{ MG-MAX-CTRL-STK-SIZE,} \\
& \text{ MG-MAX-TEMP-STK-SIZE,} \\
& \text{ MG-WORD-SIZE,} \\
& \text{ 'run))))) \\
= & \text{ p-state (tag ('pc, cons (subr, length (code (cinfo)) + 5)),} \\
& \hspace{10em} \text{ctrl-stk,} \\
& \text{ map-down-values (mg-alist (mg-state),} \\
& \hspace{10em} \text{bindings (top (ctrl-stk)),} \\
& \hspace{10em} \text{temp-stk}),
\end{aligned}$$

```

translate-proc-list (proc-list),
list (list ('c-c, '(nat 1))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-index-array-step-13-no-error

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
                               proc-list,
                               list (length (temp-stk),
                                       p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-index-array)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp (proc-list)
 ∧ ok-mg-statep (mg-state, r-cond-list)
 ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
      = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                code2))
 ∧ user-defined-procp (subr, proc-list)
 ∧ listp (ctrl-stk)
 ∧ all-cars-unique (mg-alist (mg-state))
 ∧ signatures-match (mg-alist (mg-state), name-alist)
 ∧ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                               bindings (top (ctrl-stk)),
                               temp-stk)
 ∧ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 ∧ normal (mg-state)
 ∧ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                                mg-alist (mg-state)))))))
 ∧ (idifference (caddr (call-actuals (stmt)),
                 untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                                mg-alist (mg-state)))))) ≠ 0))
 → (p-step (p-state (tag ('pc, '(mg-index-array . 7)),
                    push (p-frame (list (cons ('ans,
                                                value (car (call-actuals (stmt)),
                                                                bindings (top (ctrl-stk)))),
                                                cons ('a,
                                                    value (cadr (call-actuals (stmt)),
                                                                bindings (top (ctrl-stk)))),
                                                cons ('i,
                                                    value (caddr (call-actuals (stmt)),

```

```

bindings (top (ctrl-stk))),
cons ('array-size,
      tag ('int,
            caddr (call-actuals (stmt))),
cons ('temp-i,
      mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                              mg-alist (mg-state)))))),
tag ('pc,
     cons (subr, length (code (cinfo))
           + 5))),
ctrl-stk),
push (tag ('int,
           idifference (caddr (call-actuals (stmt)),
                        untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                                    mg-alist (mg-state)))))),
map-down-values (mg-alist (mg-state),
                  bindings (top (ctrl-stk)),
                  temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
= p-state (tag ('pc, ' (mg-index-array . 8)),
           push (p-frame (list (cons ('ans,
                                      value (car (call-actuals (stmt)),
                                                bindings (top (ctrl-stk)))),
                                cons ('a,
                                      value (cadr (call-actuals (stmt)),
                                                bindings (top (ctrl-stk)))),
                                cons ('i,
                                      value (caddr (call-actuals (stmt)),
                                                bindings (top (ctrl-stk)))),
                                cons ('array-size,
                                      tag ('int,
                                            caddr (call-actuals (stmt)))),
                                cons ('temp-i,
                                      mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                                    mg-alist (mg-state)))))),
tag ('pc,
     cons (subr, length (code (cinfo))
           + 5))),

```

```

      ctrl-stk),
    map-down-values (mg-alist (mg-state),
                      bindings (top (ctrl-stk)),
                      temp-stk),
    translate-proc-list (proc-list),
    list (list ('c-c,
               mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
    MG-MAX-CTRL-STK-SIZE,
    MG-MAX-TEMP-STK-SIZE,
    MG-WORD-SIZE,
    'run))

```

THEOREM: mg-index-array-steps-14-15-no-error

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
                             proc-list,
                             list (length (temp-stk),
                                     p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-index-array)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp (proc-list)
 ∧ ok-mg-statep (mg-state, r-cond-list)
 ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
      = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                code2))
 ∧ user-defined-procp (subr, proc-list)
 ∧ listp (ctrl-stk)
 ∧ all-cars-unique (mg-alist (mg-state))
 ∧ signatures-match (mg-alist (mg-state), name-alist)
 ∧ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                               bindings (top (ctrl-stk)),
                               temp-stk)
 ∧ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 ∧ normal (mg-state)
 ∧ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                             mg-alist (mg-state)))))))
 ∧ (idifference (caddr (call-actuals (stmt)),
                untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                             mg-alist (mg-state))))))) ≠ 0))
→ (p-step (p-step (p-state (tag ('pc, ' (mg-index-array . 8)),
                             push (p-frame (list (cons ('ans,
                                                         value (car (call-actuals (stmt)),
                                                         bindings (top (ctrl-stk))))),

```

```

cons ('a,
      value (cadr (call-actuals (stmt)),
              bindings (top (ctrl-stk)))),
cons ('i,
      value (caddr (call-actuals (stmt)),
              bindings (top (ctrl-stk)))),
cons ('array-size,
      tag ('int,
            caddr (call-actuals (stmt)))),
cons ('temp-i,
      mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                              mg-alist (mg-state))),
                                tag ('pc,
                                      cons (subr,
                                              length (code (cinfo))
                                              + 5))),
ctrl-stk),
map-down-values (mg-alist (mg-state),
                  bindings (top (ctrl-stk)),
                  temp-stk),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-state),
                                t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run)))
= p-state (tag ('pc, '(mg-index-array . 10)),
           push (p-frame (list (cons ('ans,
                                      value (car (call-actuals (stmt)),
                                              bindings (top (ctrl-stk)))),
cons ('a,
      value (cadr (call-actuals (stmt)),
              bindings (top (ctrl-stk)))),
cons ('i,
      value (caddr (call-actuals (stmt)),
              bindings (top (ctrl-stk)))),
cons ('array-size,
      tag ('int,
            caddr (call-actuals (stmt)))),
cons ('temp-i,
      mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                              mg-alist (mg-state))))),

```

```

tag('pc,
    cons(subr, length(code(cinfo)
                        + 5))),
ctrl-stk),
push(mg-to-p-simple-literal(caddr(assoc(caddr(call-actuals(stmt)),
                                         mg-alist(mg-state)))))
push(value(cadr(call-actuals(stmt)),
            bindings(top(ctrl-stk))),
      map-down-values(mg-alist(mg-state),
                      bindings(top(ctrl-stk)),
                      temp-stk))),
translate-proc-list(proc-list),
list(list('c-c,
          mg-cond-to-p-nat(cc(mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-index-array-step-16-no-error

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep(stmt,
                             proc-list,
                             list(length(temp-stk),
                                     p-ctrl-stk-size(ctrl-stk))))
 ∧ (car(stmt) = 'predefined-proc-call-mg)
 ∧ (call-name(stmt) = 'mg-index-array)
 ∧ ok-mg-statement(stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp(proc-list)
 ∧ ok-mg-statep(mg-state, r-cond-list)
 ∧ (code(translate-def-body(assoc(subr, proc-list), proc-list))
    = append(code(translate(cinfo, t-cond-list, stmt, proc-list)),
              code2))
 ∧ user-defined-procp(subr, proc-list)
 ∧ listp(ctrl-stk)
 ∧ all-cars-unique(mg-alist(mg-state))
 ∧ signatures-match(mg-alist(mg-state), name-alist)
 ∧ mg-vars-list-ok-in-p-state(mg-alist(mg-state),
                               bindings(top(ctrl-stk)),
                               temp-stk)
 ∧ no-p-aliasing(bindings(top(ctrl-stk)), mg-alist(mg-state))
 ∧ normal(mg-state)
 ∧ (¬ negativep(untag(mg-to-p-simple-literal(caddr(assoc(caddr(call-actuals(stmt)),
                                                         mg-alist(mg-state)))))))

```

$$\begin{aligned}
& \wedge \quad (\text{idifference}(\text{caddr}(\text{call-actuals}(stmt)), \\
& \quad \text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(stmt)), \\
& \quad \quad \quad \text{mg-alist}(mg-state)))))) \neq 0)) \\
\rightarrow & \quad (\text{p-step}(\text{p-state}(\text{tag}('pc, '(mg-index-array . 10)), \\
& \quad \text{push}(\text{p-frame}(\text{list}(\text{cons}('ans, \\
& \quad \quad \quad \text{value}(\text{car}(\text{call-actuals}(stmt)), \\
& \quad \quad \quad \text{bindings}(\text{top}(ctrl-stk)))), \\
& \quad \text{cons}('a, \\
& \quad \quad \text{value}(\text{cadr}(\text{call-actuals}(stmt)), \\
& \quad \quad \text{bindings}(\text{top}(ctrl-stk)))), \\
& \quad \text{cons}('i, \\
& \quad \quad \text{value}(\text{caddr}(\text{call-actuals}(stmt)), \\
& \quad \quad \text{bindings}(\text{top}(ctrl-stk)))), \\
& \quad \text{cons}('array-size, \\
& \quad \quad \text{tag}('int, \\
& \quad \quad \quad \text{caddr}(\text{call-actuals}(stmt))), \\
& \quad \text{cons}('temp-i, \\
& \quad \quad \text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(stmt)), \\
& \quad \quad \quad \text{mg-alist}(mg-state)))))), \\
& \quad \text{tag}('pc, \\
& \quad \quad \text{cons}(subr, \text{length}(\text{code}(cinfo)) \\
& \quad \quad \quad + 5))), \\
& \quad ctrl-stk), \\
& \text{push}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(stmt)), \\
& \quad \quad \quad \text{mg-alist}(mg-state))))), \\
& \quad \text{push}(\text{value}(\text{cadr}(\text{call-actuals}(stmt)), \\
& \quad \quad \text{bindings}(\text{top}(ctrl-stk))), \\
& \quad \text{map-down-values}(\text{mg-alist}(mg-state), \\
& \quad \quad \text{bindings}(\text{top}(ctrl-stk)), \\
& \quad \quad \quad temp-stk))), \\
& \text{translate-proc-list}(proc-list), \\
& \text{list}(\text{list}('c-c, \\
& \quad \text{mg-cond-to-p-nat}(\text{cc}(mg-state), t-cond-list))), \\
& \text{MG-MAX-CTRL-STK-SIZE}, \\
& \text{MG-MAX-TEMP-STK-SIZE}, \\
& \text{MG-WORD-SIZE}, \\
& 'run)) \\
= & \quad \text{p-state}(\text{tag}('pc, '(mg-index-array . 11)), \\
& \quad \text{push}(\text{p-frame}(\text{list}(\text{cons}('ans, \\
& \quad \quad \quad \text{value}(\text{car}(\text{call-actuals}(stmt)), \\
& \quad \quad \quad \text{bindings}(\text{top}(ctrl-stk))), \\
& \quad \text{cons}('a, \\
& \quad \quad \text{value}(\text{cadr}(\text{call-actuals}(stmt)), \\
& \quad \quad \text{bindings}(\text{top}(ctrl-stk))),
\end{aligned}$$

```

cons('i,
    value(caddr(call-actuals(stmt)),
          bindings(top(ctrl-stk)))),
cons('array-size,
    tag('int,
        caddr(call-actuals(stmt))),
cons('temp-i,
    mg-to-p-simple-literal(caddr(assoc(caddr(call-actuals(stmt)),
                                       mg-alist(mg-state)))))),
tag('pc,
    cons(subr, length(code(cinfo))
          + 5)),
ctrl-stk),
push(tag('nat,
    untag(mg-to-p-simple-literal(caddr(assoc(caddr(call-actuals(stmt)),
                                             mg-alist(mg-state)))))),
    push(value(cadr(call-actuals(stmt)),
              bindings(top(ctrl-stk))),
          map-down-values(mg-alist(mg-state),
                          bindings(top(ctrl-stk)),
                          temp-stk))),
translate-proc-list(proc-list),
list(list('c-c,
    mg-cond-to-p-nat(cc(mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: non-negative-integerp-small-naturalp

```

(integerp(x)
  ∧ (¬ negativep(x))
  ∧ small-integerp(y, n)
  ∧ (y ∈ ℕ)
  ∧ (idifference(y, x) ≠ 0))
→ small-naturalp(x, n)

```

THEOREM: mg-index-array-step-17-no-error

```

((n ≠ 0)
  ∧ (¬ resources-inadequatep(stmt,
                               proc-list,
                               list(length(temp-stk),
                                       p-ctrl-stk-size(ctrl-stk))))
  ∧ (car(stmt) = 'predefined-proc-call-mg)

```

```

^ (call-name (stmt) = 'mg-index-array)
^ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
^ ok-mg-def-plistp (proc-list)
^ ok-mg-statep (mg-state, r-cond-list)
^ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
              code2))
^ user-defined-procp (subr, proc-list)
^ listp (ctrl-stk)
^ all-cars-unique (mg-alist (mg-state))
^ signatures-match (mg-alist (mg-state), name-alist)
^ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                             bindings (top (ctrl-stk)),
                             temp-stk)
^ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
^ normal (mg-state)
^ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                            mg-alist (mg-state)))))))
^ (idifference (caddr (call-actuals (stmt)),
               untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                            mg-alist (mg-state))))))) ≠ 0))
→ (p-step (p-state (tag ('pc, ' (mg-index-array . 11)),
                    push (p-frame (list (cons ('ans,
                                              value (car (call-actuals (stmt)),
                                              bindings (top (ctrl-stk)))),
                                          cons ('a,
                                              value (cadr (call-actuals (stmt)),
                                              bindings (top (ctrl-stk)))),
                                          cons ('i,
                                              value (caddr (call-actuals (stmt)),
                                              bindings (top (ctrl-stk)))),
                                          cons ('array-size,
                                              tag ('int,
                                                  caddr (call-actuals (stmt)))),
                                          cons ('temp-i,
                                              mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                                 mg-alist (mg-state))))))),
                                              tag ('pc,
                                                  cons (subr, length (code (cinfo))
                                                  + 5))),
                    ctrl-stk),
    push (tag ('nat,
              untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                            mg-alist (mg-state))))))),

```

```

push (value (cadr (call-actuals (stmt)),
      bindings (top (ctrl-stk))),
      map-down-values (mg-alist (mg-state),
      bindings (top (ctrl-stk)),
      temp-stk))),
translate-proc-list (proc-list),
list (list ('c-c,
      mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
= p-state (tag ('pc, ' (mg-index-array . 12)),
push (p-frame (list (cons ('ans,
      value (car (call-actuals (stmt)),
      bindings (top (ctrl-stk))),
cons ('a,
      value (cadr (call-actuals (stmt)),
      bindings (top (ctrl-stk))),
cons ('i,
      value (caddr (call-actuals (stmt)),
      bindings (top (ctrl-stk))),
cons ('array-size,
      tag ('int,
      caddr (call-actuals (stmt))),
cons ('temp-i,
      mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
      mg-alist (mg-state)))))),
tag ('pc,
      cons (subr, length (code (cinfo))
      + 5))),
ctrl-stk),
push (tag ('nat,
      untag (value (cadr (call-actuals (stmt)),
      bindings (top (ctrl-stk))))
      + untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
      mg-alist (mg-state)))))),
      map-down-values (mg-alist (mg-state),
      bindings (top (ctrl-stk)),
      temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
      mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,

```

MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: mg-index-array-index-lessp-temp-stk-length

((car (stmt) = 'predefined-proc-call-mg)
 $\wedge$ (call-name (stmt) = 'mg-index-array)
 $\wedge$ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 $\wedge$ ok-mg-statep (mg-state, r-cond-list)
 $\wedge$ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
bindings (top (ctrl-stk)),
temp-stk)
 $\wedge$ signatures-match (mg-alist (mg-state), name-alist)
 $\wedge$ ($\neg$ negativep (untag (caddr (assoc (caddr (call-actuals (stmt)),
mg-alist (mg-state))))))
 $\wedge$ (idifference (caddr (call-actuals (stmt)),
untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
mg-alist (mg-state)))))) $\neq$ 0))
 $\rightarrow$ (((untag (value (cadr (call-actuals (stmt)), bindings (top (ctrl-stk))))
+ untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
mg-alist (mg-state))))))
< length (temp-stk))
= t)

THEOREM: mg-index-array-step-18-no-error

((n $\neq$ 0)
 $\wedge$ ($\neg$ resources-inadequatep (stmt,
proc-list,
list (length (temp-stk),
p-ctrl-stk-size (ctrl-stk))))
 $\wedge$ (car (stmt) = 'predefined-proc-call-mg)
 $\wedge$ (call-name (stmt) = 'mg-index-array)
 $\wedge$ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 $\wedge$ ok-mg-def-plistp (proc-list)
 $\wedge$ ok-mg-statep (mg-state, r-cond-list)
 $\wedge$ (code (translate-def-body (assoc (subr, proc-list), proc-list))
= append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
code2))
 $\wedge$ user-defined-procp (subr, proc-list)
 $\wedge$ listp (ctrl-stk)
 $\wedge$ all-cars-unique (mg-alist (mg-state))
 $\wedge$ signatures-match (mg-alist (mg-state), name-alist)
 $\wedge$ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
bindings (top (ctrl-stk)),

$$\begin{aligned}
& \text{temp-stk}) \\
\wedge & \text{no-p-aliasing}(\text{bindings}(\text{top}(\text{ctrl-stk})), \text{mg-alist}(\text{mg-state})) \\
\wedge & \text{normal}(\text{mg-state}) \\
\wedge & (\neg \text{negativep}(\text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(\text{stmt})), \\
& \text{mg-alist}(\text{mg-state}))))))) \\
\wedge & (\text{idifference}(\text{caddr}(\text{call-actuals}(\text{stmt})), \\
& \text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(\text{stmt})), \\
& \text{mg-alist}(\text{mg-state})))))) \neq 0)) \\
\rightarrow & (\text{p-step}(\text{p-state}(\text{tag}('pc, '(mg-index-array . 12)), \\
& \text{push}(\text{p-frame}(\text{list}(\text{cons}('ans, \\
& \text{value}(\text{car}(\text{call-actuals}(\text{stmt})), \\
& \text{bindings}(\text{top}(\text{ctrl-stk}))), \\
& \text{cons}('a, \\
& \text{value}(\text{cadr}(\text{call-actuals}(\text{stmt})), \\
& \text{bindings}(\text{top}(\text{ctrl-stk}))), \\
& \text{cons}('i, \\
& \text{value}(\text{caddr}(\text{call-actuals}(\text{stmt})), \\
& \text{bindings}(\text{top}(\text{ctrl-stk}))), \\
& \text{cons}('array-size, \\
& \text{tag}('int, \\
& \text{caddr}(\text{call-actuals}(\text{stmt}))), \\
& \text{cons}('temp-i, \\
& \text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(\text{stmt})), \\
& \text{mg-alist}(\text{mg-state})))))), \\
& \text{tag}('pc, \\
& \text{cons}(\text{subr}, \text{length}(\text{code}(\text{cinfo}) \\
& + 5))), \\
& \text{ctrl-stk}), \\
& \text{push}(\text{tag}('nat, \\
& \text{untag}(\text{value}(\text{cadr}(\text{call-actuals}(\text{stmt})), \\
& \text{bindings}(\text{top}(\text{ctrl-stk})))) \\
& + \text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(\text{stmt})), \\
& \text{mg-alist}(\text{mg-state})))))), \\
& \text{map-down-values}(\text{mg-alist}(\text{mg-state}), \\
& \text{bindings}(\text{top}(\text{ctrl-stk})), \\
& \text{temp-stk}), \\
& \text{translate-proc-list}(\text{proc-list}), \\
& \text{list}(\text{list}('c-c, \\
& \text{mg-cond-to-p-nat}(\text{cc}(\text{mg-state}), \text{t-cond-list}))), \\
& \text{MG-MAX-CTRL-STK-SIZE}, \\
& \text{MG-MAX-TEMP-STK-SIZE}, \\
& \text{MG-WORD-SIZE}, \\
& 'run)) \\
= & \text{p-state}(\text{tag}('pc, '(mg-index-array . 13)),
\end{aligned}$$

```

push (p-frame (list (cons ('ans,
                           value (car (call-actuals (stmt)),
                           bindings (top (ctrl-stk)))))
                 cons ('a,
                       value (cadr (call-actuals (stmt)),
                       bindings (top (ctrl-stk)))))
                 cons ('i,
                       value (caddr (call-actuals (stmt)),
                       bindings (top (ctrl-stk)))))
                 cons ('array-size,
                       tag ('int,
                           caddr (call-actuals (stmt)))))
                 cons ('temp-i,
                       mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                                mg-alist (mg-state)))))),
          tag ('pc,
              cons (subr, length (code (cinfo)
                                     + 5))),
          ctrl-stk),
push (rget (untag (value (cadr (call-actuals (stmt)),
                           bindings (top (ctrl-stk)))))
      + untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                                mg-alist (mg-state)))))),
      map-down-values (mg-alist (mg-state),
                        bindings (top (ctrl-stk)),
                        temp-stk)),
map-down-values (mg-alist (mg-state),
                bindings (top (ctrl-stk)),
                temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-index-array-step-19-no-error

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge (\neg \text{resources-inadequatep} (stmt, \\
& \quad \text{proc-list}, \\
& \quad \text{list (length (temp-stk),} \\
& \quad \quad \text{p-ctrl-stk-size (ctrl-stk))})) \\
& \wedge (\text{car (stmt)} = \text{'predefined-proc-call-mg})
\end{aligned}$$

```

^ (call-name (stmt) = 'mg-index-array)
^ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
^ ok-mg-def-plistp (proc-list)
^ ok-mg-statep (mg-state, r-cond-list)
^ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
        code2))
^ user-defined-procp (subr, proc-list)
^ listp (ctrl-stk)
^ all-cars-unique (mg-alist (mg-state))
^ signatures-match (mg-alist (mg-state), name-alist)
^ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
    bindings (top (ctrl-stk)),
    temp-stk)
^ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
^ normal (mg-state)
^ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
    mg-alist (mg-state)))))))
^ (idifference (caddr (call-actuals (stmt)),
    untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
    mg-alist (mg-state)))))) ≠ 0))
→ (p-step (p-state (tag ('pc, '(mg-index-array . 13)),
    push (p-frame (list (cons ('ans,
        value (car (call-actuals (stmt)),
        bindings (top (ctrl-stk)))),
    cons ('a,
        value (cadr (call-actuals (stmt)),
        bindings (top (ctrl-stk)))),
    cons ('i,
        value (caddr (call-actuals (stmt)),
        bindings (top (ctrl-stk)))),
    cons ('array-size,
        tag ('int,
            caddr (call-actuals (stmt)))),
    cons ('temp-i,
        mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
            mg-alist (mg-state)))))),
    tag ('pc,
        cons (subr, length (code (cinfo))
            + 5))),
    ctrl-stk),
    push (rget (untag (value (cadr (call-actuals (stmt)),
        bindings (top (ctrl-stk))))
    + untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),

```

```

map-down-values (mg-alist (mg-state),
                   bindings (top (ctrl-stk)),
                   temp-stk)),
map-down-values (mg-alist (mg-state),
                   bindings (top (ctrl-stk)),
                   temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
= p-state (tag ('pc, ' (mg-index-array . 14)),
           push (p-frame (list (cons ('ans,
                                     value (car (call-actuals (stmt)),
                                     bindings (top (ctrl-stk)))))
                               cons ('a,
                                     value (cadr (call-actuals (stmt)),
                                     bindings (top (ctrl-stk)))))
                               cons ('i,
                                     value (caddr (call-actuals (stmt)),
                                     bindings (top (ctrl-stk)))))
                               cons ('array-size,
                                     tag ('int,
                                     caddr (call-actuals (stmt)))))
                               cons ('temp-i,
                                     mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                     mg-alist (mg-state))))))
                               tag ('pc,
                                   cons (subr, length (code (cinfo))
                                   + 5))),
ctrl-stk),
push (value (car (call-actuals (stmt)),
            bindings (top (ctrl-stk))),
push (rget (untag (value (cadr (call-actuals (stmt)),
                        bindings (top (ctrl-stk)))))
+ untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
mg-alist (mg-state)))))),
map-down-values (mg-alist (mg-state),
                  bindings (top (ctrl-stk)),
                  temp-stk)),
map-down-values (mg-alist (mg-state),

```

bindings (top (ctrl-stk)),
temp-stk))),

translate-proc-list (proc-list),
list (list ('c-c,
mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: mg-index-array-step-20-no-error

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
 proc-list,
 list (length (temp-stk),
 p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-index-array)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp (proc-list)
 ∧ ok-mg-statep (mg-state, r-cond-list)
 ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
 = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
 code2))
 ∧ user-defined-procp (subr, proc-list)
 ∧ listp (ctrl-stk)
 ∧ all-cars-unique (mg-alist (mg-state))
 ∧ signatures-match (mg-alist (mg-state), name-alist)
 ∧ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
 bindings (top (ctrl-stk)),
 temp-stk)
 ∧ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 ∧ normal (mg-state)
 ∧ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
 mg-alist (mg-state)))))))
 ∧ (idifference (caddr (call-actuals (stmt)),
 untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
 mg-alist (mg-state))))))) ≠ 0))
 → (p-step (p-state (tag ('pc, ' (mg-index-array . 14)),
 push (p-frame (list (cons ('ans,
 value (car (call-actuals (stmt)),
 bindings (top (ctrl-stk)))),
 cons ('a,
 value (cadr (call-actuals (stmt))),

```

bindings (top (ctrl-stk))),
cons ('i,
      value (caddr (call-actuals (stmt)),
              bindings (top (ctrl-stk))),
cons ('array-size,
      tag ('int,
            caddr (call-actuals (stmt))),
cons ('temp-i,
      mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                              mg-alist (mg-state)))))),
tag ('pc,
     cons (subr, length (code (cinfo))
           + 5))),
ctrl-stk),
push (value (car (call-actuals (stmt)),
             bindings (top (ctrl-stk))),
push (rget (untag (value (cadr (call-actuals (stmt)),
                             bindings (top (ctrl-stk))))
          + untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                         mg-alist (mg-state)))))),
      map-down-values (mg-alist (mg-state),
                        bindings (top (ctrl-stk)),
                        temp-stk)),
      map-down-values (mg-alist (mg-state),
                        bindings (top (ctrl-stk)),
                        temp-stk))),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
= p-state (tag ('pc, ' (mg-index-array . 15)),
           push (p-frame (list (cons ('ans,
                                      value (car (call-actuals (stmt)),
                                                bindings (top (ctrl-stk)))),
                                cons ('a,
                                      value (cadr (call-actuals (stmt)),
                                                bindings (top (ctrl-stk))))),
                                cons ('i,
                                      value (caddr (call-actuals (stmt)),
                                                bindings (top (ctrl-stk))))),
                                cons ('array-size,
                                      value (caddr (call-actuals (stmt)),
                                                bindings (top (ctrl-stk))))))

```

```

tag('int,
  caddr (call-actuals (stmt))),
cons('temp-i,
  mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
    mg-alist (mg-state)))))),
tag('pc,
  cons (subr, length (code (cinfo)
    + 5))),
ctrl-stk),
rput (rget (untag (value (cadr (call-actuals (stmt)),
  bindings (top (ctrl-stk))))
+ untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
  mg-alist (mg-state)))))),
  map-down-values (mg-alist (mg-state),
    bindings (top (ctrl-stk)),
    temp-stk)),
  untag (value (car (call-actuals (stmt)),
    bindings (top (ctrl-stk))))),
  map-down-values (mg-alist (mg-state),
    bindings (top (ctrl-stk)),
    temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
  mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-index-array-steps-21-22-no-error

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
  proc-list,
  list (length (temp-stk),
    p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-index-array)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp (proc-list)
 ∧ ok-mg-statement (mg-state, r-cond-list)
 ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
  = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
    code2))
 ∧ user-defined-procp (subr, proc-list)

```

```

 $\wedge$  listp (ctrl-stk)
 $\wedge$  all-cars-unique (mg-alist (mg-state))
 $\wedge$  signatures-match (mg-alist (mg-state), name-alist)
 $\wedge$  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                     bindings (top (ctrl-stk)),
                                     temp-stk)
 $\wedge$  no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 $\wedge$  normal (mg-state)
 $\wedge$  ( $\neg$  negativep (untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                                mg-alist (mg-state)))))))
 $\wedge$  (idifference (caddr (call-actuals (stmt)),
                    untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                                mg-alist (mg-state))))))  $\neq$  0))
 $\rightarrow$  (p-step (p-step (p-state (tag ('pc, '(mg-index-array . 15)),
                                push (p-frame (list (cons ('ans,
                                                            value (car (call-actuals (stmt)),
                                                            bindings (top (ctrl-stk))),
                                                            cons ('a,
                                                            value (cadr (call-actuals (stmt)),
                                                            bindings (top (ctrl-stk))),
                                                            cons ('i,
                                                            value (caddr (call-actuals (stmt)),
                                                            bindings (top (ctrl-stk))),
                                                            cons ('array-size,
                                                            tag ('int,
                                                            caddr (call-actuals (stmt))),
                                                            cons ('temp-i,
                                                            mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                                mg-alist (mg-state))))),
                                                            tag ('pc,
                                                            cons (subr,
                                                            length (code (cinfo))
                                                            + 5))),
                                ctrl-stk),
                                rput (rget (untag (value (cadr (call-actuals (stmt)),
                                                            bindings (top (ctrl-stk))))
                                + untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                                mg-alist (mg-state))))),
                                map-down-values (mg-alist (mg-state),
                                                    bindings (top (ctrl-stk)),
                                                    temp-stk)),
                                untag (value (car (call-actuals (stmt)),
                                                    bindings (top (ctrl-stk))))),
                                map-down-values (mg-alist (mg-state),

```

$$\begin{aligned}
& \text{bindings}(\text{top}(\text{ctrl-stk})), \\
& \text{temp-stk}), \\
& \text{translate-proc-list}(\text{proc-list}), \\
& \text{list}(\text{list}('c-c, \\
& \quad \text{mg-cond-to-p-nat}(\text{cc}(\text{mg-state}), \\
& \quad \quad \text{t-cond-list}))), \\
& \text{MG-MAX-CTRL-STK-SIZE}, \\
& \text{MG-MAX-TEMP-STK-SIZE}, \\
& \text{MG-WORD-SIZE}, \\
& 'run))) \\
= & \text{p-state}(\text{tag}('pc, \text{cons}(\text{subr}, \text{length}(\text{code}(\text{cinfo})) + 5)), \\
& \text{ctrl-stk}, \\
& \text{rput}(\text{rget}(\text{untag}(\text{value}(\text{cadr}(\text{call-actuals}(\text{stmt})), \\
& \quad \text{bindings}(\text{top}(\text{ctrl-stk})))) \\
& + \text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(\text{stmt})), \\
& \quad \text{mg-alist}(\text{mg-state}))))), \\
& \text{map-down-values}(\text{mg-alist}(\text{mg-state}), \\
& \quad \text{bindings}(\text{top}(\text{ctrl-stk})), \\
& \quad \text{temp-stk})), \\
& \text{untag}(\text{value}(\text{car}(\text{call-actuals}(\text{stmt})), \\
& \quad \text{bindings}(\text{top}(\text{ctrl-stk}))))), \\
& \text{map-down-values}(\text{mg-alist}(\text{mg-state}), \\
& \quad \text{bindings}(\text{top}(\text{ctrl-stk})), \\
& \quad \text{temp-stk})), \\
& \text{translate-proc-list}(\text{proc-list}), \\
& \text{list}(\text{list}('c-c, \\
& \quad \text{mg-cond-to-p-nat}(\text{cc}(\text{mg-state}), \text{t-cond-list}))), \\
& \text{MG-MAX-CTRL-STK-SIZE}, \\
& \text{MG-MAX-TEMP-STK-SIZE}, \\
& \text{MG-WORD-SIZE}, \\
& 'run))
\end{aligned}$$

THEOREM: mg-index-array-push-cc

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge (\neg \text{resources-inadequatep}(\text{stmt}, \\
& \quad \text{proc-list}, \\
& \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \quad \text{p-ctrl-stk-size}(\text{ctrl-stk})))) \\
& \wedge (\text{car}(\text{stmt}) = \text{'predefined-proc-call-mg}) \\
& \wedge (\text{call-name}(\text{stmt}) = \text{'mg-index-array}) \\
& \wedge \text{ok-mg-statement}(\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}) \\
& \wedge \text{ok-mg-def-plistp}(\text{proc-list}) \\
& \wedge \text{ok-mg-statement}(\text{mg-state}, \text{r-cond-list}) \\
& \wedge (\text{code}(\text{translate-def-body}(\text{assoc}(\text{subr}, \text{proc-list}), \text{proc-list}))
\end{aligned}$$

$$\begin{aligned}
&= \text{append}(\text{code}(\text{translate}(cinfo, t\text{-cond-list}, stmt, proc\text{-list})), \\
&\quad code2)) \\
\wedge &\text{user-defined-procp}(subr, proc\text{-list}) \\
\wedge &\text{listp}(ctrl\text{-stk}) \\
\wedge &\text{all-cars-unique}(\text{mg-alist}(mg\text{-state})) \\
\wedge &\text{signatures-match}(\text{mg-alist}(mg\text{-state}), name\text{-alist}) \\
\wedge &\text{normal}(mg\text{-state}) \\
\rightarrow &(\text{p-step}(\text{p-state}(\text{tag}('pc, \text{cons}(subr, \text{length}(\text{code}(cinfo)) + 5)), \\
&\quad ctrl\text{-stk}, \\
&\quad temp\text{-stk}, \\
&\quad \text{translate-proc-list}(proc\text{-list}), \\
&\quad \text{list}(\text{list}('c-c, cc\text{-value})), \\
&\quad \text{MG-MAX-CTRL-STK-SIZE}, \\
&\quad \text{MG-MAX-TEMP-STK-SIZE}, \\
&\quad \text{MG-WORD-SIZE}, \\
&\quad 'run)) \\
&= \text{p-state}(\text{tag}('pc, \text{cons}(subr, \text{length}(\text{code}(cinfo)) + 6)), \\
&\quad ctrl\text{-stk}, \\
&\quad \text{push}(cc\text{-value}, temp\text{-stk}), \\
&\quad \text{translate-proc-list}(proc\text{-list}), \\
&\quad \text{list}(\text{list}('c-c, cc\text{-value})), \\
&\quad \text{MG-MAX-CTRL-STK-SIZE}, \\
&\quad \text{MG-MAX-TEMP-STK-SIZE}, \\
&\quad \text{MG-WORD-SIZE}, \\
&\quad 'run))
\end{aligned}$$

THEOREM: mg-index-array-sub1-cc

$$\begin{aligned}
&((n \neq 0) \\
\wedge &(\neg \text{resources-inadequatep}(stmt, \\
&\quad proc\text{-list}, \\
&\quad \text{list}(\text{length}(temp\text{-stk}), \\
&\quad \text{p-ctrl-stk-size}(ctrl\text{-stk})))) \\
\wedge &(\text{car}(stmt) = 'predefined-proc-call-mg) \\
\wedge &(\text{call-name}(stmt) = 'mg-index-array) \\
\wedge &\text{ok-mg-statement}(stmt, r\text{-cond-list}, name\text{-alist}, proc\text{-list}) \\
\wedge &\text{ok-mg-def-plistp}(proc\text{-list}) \\
\wedge &\text{ok-mg-statep}(mg\text{-state}, r\text{-cond-list}) \\
\wedge &(\text{code}(\text{translate-def-body}(\text{assoc}(subr, proc\text{-list}), proc\text{-list})) \\
&\quad = \text{append}(\text{code}(\text{translate}(cinfo, t\text{-cond-list}, stmt, proc\text{-list})), \\
&\quad code2)) \\
\wedge &\text{user-defined-procp}(subr, proc\text{-list}) \\
\wedge &\text{listp}(ctrl\text{-stk}) \\
\wedge &\text{all-cars-unique}(\text{mg-alist}(mg\text{-state})) \\
\wedge &\text{signatures-match}(\text{mg-alist}(mg\text{-state}), name\text{-alist})
\end{aligned}$$

$$\begin{aligned}
& \wedge \text{normal}(mg\text{-state}) \\
& \wedge (cc\text{-value} \in \text{list}('(\text{nat } 1), '(\text{nat } 2)))) \\
\rightarrow & (\text{p-step}(\text{p-state}(\text{tag}('pc, \text{cons}(subr, \text{length}(\text{code}(cinfo)) + 6)), \\
& \quad ctrl\text{-stk}, \\
& \quad \text{push}(cc\text{-value}, temp\text{-stk}), \\
& \quad \text{translate-proc-list}(proc\text{-list}), \\
& \quad \text{list}(\text{list}('c-c, cc\text{-value})), \\
& \quad \text{MG-MAX-CTRL-STK-SIZE}, \\
& \quad \text{MG-MAX-TEMP-STK-SIZE}, \\
& \quad \text{MG-WORD-SIZE}, \\
& \quad 'run)) \\
= & \text{p-state}(\text{tag}('pc, \text{cons}(subr, \text{length}(\text{code}(cinfo)) + 7)), \\
& \quad ctrl\text{-stk}, \\
& \quad \text{push}(\text{tag}('nat, \text{untag}(cc\text{-value}) - 1), temp\text{-stk}), \\
& \quad \text{translate-proc-list}(proc\text{-list}), \\
& \quad \text{list}(\text{list}('c-c, cc\text{-value})), \\
& \quad \text{MG-MAX-CTRL-STK-SIZE}, \\
& \quad \text{MG-MAX-TEMP-STK-SIZE}, \\
& \quad \text{MG-WORD-SIZE}, \\
& \quad 'run))
\end{aligned}$$

THEOREM: mg-index-array-last-step-error-case

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge (\neg \text{resources-inadequatep}(stmt, \\
& \quad \quad \quad proc\text{-list}, \\
& \quad \quad \quad \text{list}(\text{length}(temp\text{-stk}), \\
& \quad \quad \quad \text{p-ctrl-stk-size}(ctrl\text{-stk})))) \\
& \wedge (\text{car}(stmt) = 'predefined-proc-call-mg) \\
& \wedge (\text{call-name}(stmt) = 'mg-index-array) \\
& \wedge \text{ok-mg-statement}(stmt, r\text{-cond-list}, name\text{-alist}, proc\text{-list}) \\
& \wedge \text{ok-mg-def-plistp}(proc\text{-list}) \\
& \wedge \text{ok-mg-statep}(mg\text{-state}, r\text{-cond-list}) \\
& \wedge (\text{code}(\text{translate-def-body}(\text{assoc}(subr, proc\text{-list}), proc\text{-list})) \\
& \quad = \text{append}(\text{code}(\text{translate}(cinfo, t\text{-cond-list}, stmt, proc\text{-list})), \\
& \quad \quad \quad code2)) \\
& \wedge \text{user-defined-procp}(subr, proc\text{-list}) \\
& \wedge \text{listp}(ctrl\text{-stk}) \\
& \wedge \text{all-cars-unique}(mg\text{-alist}(mg\text{-state})) \\
& \wedge \text{signatures-match}(mg\text{-alist}(mg\text{-state}), name\text{-alist}) \\
& \wedge \text{mg-vars-list-ok-in-p-state}(mg\text{-alist}(mg\text{-state}), \\
& \quad \quad \quad \text{bindings}(\text{top}(ctrl\text{-stk}), \\
& \quad \quad \quad temp\text{-stk})) \\
& \wedge \text{no-p-aliasing}(\text{bindings}(\text{top}(ctrl\text{-stk}), mg\text{-alist}(mg\text{-state}))) \\
& \wedge \text{normal}(mg\text{-state})
\end{aligned}$$

$$\begin{aligned}
& \wedge \quad (\text{negativep}(\text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(stmt)), \\
& \hspace{15em} \text{mg-alist}(mg-state)))))) \\
& \vee \quad (\text{idifference}(\text{caddr}(\text{call-actuals}(stmt)), \\
& \hspace{10em} \text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(stmt)), \\
& \hspace{15em} \text{mg-alist}(mg-state)))))) \simeq 0))) \\
\rightarrow & \quad (\text{p-step}(\text{p-state}(\text{tag}('pc, \text{cons}(subr, \text{length}(\text{code}(cinfo)) + 7)), \\
& \hspace{10em} ctrl-stk, \\
& \hspace{10em} \text{push}(\text{tag}('nat, \text{untag}('nat 1) - 1), \\
& \hspace{15em} \text{map-down-values}(\text{mg-alist}(mg-state), \\
& \hspace{15em} \text{bindings}(\text{top}(ctrl-stk)), \\
& \hspace{15em} temp-stk)), \\
& \hspace{10em} \text{translate-proc-list}(proc-list), \\
& \hspace{10em} '((c-c (nat 1))), \\
& \hspace{10em} \text{MG-MAX-CTRL-STK-SIZE}, \\
& \hspace{10em} \text{MG-MAX-TEMP-STK-SIZE}, \\
& \hspace{10em} \text{MG-WORD-SIZE}, \\
& \hspace{10em} 'run)) \\
= & \quad \text{p-state}(\text{tag}('pc, \\
& \hspace{10em} \text{cons}(subr, \\
& \hspace{15em} \text{if normal}(\text{mg-meaning-r}(stmt, \\
& \hspace{15em} proc-list, \\
& \hspace{15em} mg-state, \\
& \hspace{15em} n, \\
& \hspace{15em} \text{list}(\text{length}(temp-stk), \\
& \hspace{15em} \text{p-ctrl-stk-size}(ctrl-stk)))) \\
& \hspace{10em} \text{then length}(\text{code}(\text{translate}(cinfo, \\
& \hspace{15em} t-cond-list, \\
& \hspace{15em} stmt, \\
& \hspace{15em} proc-list))) \\
& \hspace{10em} \text{else find-label}(\text{fetch-label}(\text{cc}(\text{mg-meaning-r}(stmt, \\
& \hspace{15em} proc-list, \\
& \hspace{15em} mg-state, \\
& \hspace{15em} n, \\
& \hspace{15em} \text{list}(\text{length}(temp-stk), \\
& \hspace{15em} \text{p-ctrl-stk-size}(ctrl-stk))))), \\
& \hspace{10em} \text{label-alist}(\text{translate}(cinfo, \\
& \hspace{15em} t-cond-list, \\
& \hspace{15em} stmt, \\
& \hspace{15em} proc-list))), \\
& \hspace{10em} \text{append}(\text{code}(\text{translate}(cinfo, \\
& \hspace{15em} t-cond-list, \\
& \hspace{15em} stmt, \\
& \hspace{15em} proc-list)), \\
& \hspace{10em} code2)) \text{endif}),
\end{aligned}$$

```

ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (stmt,
                                         proc-list,
                                         mg-state,
                                         n,
                                         list (length (temp-stk),
                                                p-ctrl-stk-size (ctrl-stk)))))
                                         bindings (top (ctrl-stk)),
                                         temp-stk),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-meaning-r (stmt,
                                                proc-list,
                                                mg-state,
                                                n,
                                                list (length (temp-stk),
                                                        p-ctrl-stk-size (ctrl-stk)))))
                                t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: simple-typed-identifier-type-equivalence

```

simple-typed-identifierp (b, type, alist)
→ ((cadr (assoc (b, alist)) = type) = t)

```

THEOREM: mg-index-array-step-25-no-error

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
                              proc-list,
                              list (length (temp-stk),
                                      p-ctrl-stk-size (ctrl-stk)))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-index-array)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp (proc-list)
 ∧ ok-mg-statep (mg-state, r-cond-list)
 ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
              code2))
 ∧ user-defined-procp (subr, proc-list)
 ∧ listp (ctrl-stk)
 ∧ all-cars-unique (mg-alist (mg-state))

```

$$\begin{aligned}
& \wedge \text{signatures-match}(\text{mg-alist}(\text{mg-state}), \text{name-alist}) \\
& \wedge \text{mg-vars-list-ok-in-p-state}(\text{mg-alist}(\text{mg-state}), \\
& \quad \text{bindings}(\text{top}(\text{ctrl-stk})), \\
& \quad \text{temp-stk}) \\
& \wedge \text{no-p-aliasing}(\text{bindings}(\text{top}(\text{ctrl-stk})), \text{mg-alist}(\text{mg-state})) \\
& \wedge \text{normal}(\text{mg-state}) \\
& \wedge (\neg \text{negativep}(\text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(\text{stmt})), \\
& \quad \text{mg-alist}(\text{mg-state}))))))) \\
& \wedge (\text{idifference}(\text{caddr}(\text{call-actuals}(\text{stmt})), \\
& \quad \text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(\text{stmt})), \\
& \quad \text{mg-alist}(\text{mg-state})))))) \neq 0)) \\
\rightarrow & (\text{p-step}(\text{p-state}(\text{tag}('pc, \text{cons}(\text{subr}, \text{length}(\text{code}(\text{cinfo})) + 7)), \\
& \quad \text{ctrl-stk}, \\
& \quad \text{push}(\text{tag}('nat, \\
& \quad \quad \text{untag}(\text{mg-cond-to-p-nat}(\text{cc}(\text{mg-state}), \\
& \quad \quad \quad \text{t-cond-list})) - 1), \\
& \quad \text{rput}(\text{rget}(\text{untag}(\text{value}(\text{cadr}(\text{call-actuals}(\text{stmt})), \\
& \quad \quad \text{bindings}(\text{top}(\text{ctrl-stk})))) \\
& \quad \quad + \text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(\text{stmt}), \\
& \quad \quad \quad \text{mg-alist}(\text{mg-state}))))), \\
& \quad \quad \text{map-down-values}(\text{mg-alist}(\text{mg-state}), \\
& \quad \quad \quad \text{bindings}(\text{top}(\text{ctrl-stk})), \\
& \quad \quad \quad \text{temp-stk})), \\
& \quad \text{untag}(\text{value}(\text{car}(\text{call-actuals}(\text{stmt})), \\
& \quad \quad \text{bindings}(\text{top}(\text{ctrl-stk}))), \\
& \quad \text{map-down-values}(\text{mg-alist}(\text{mg-state}), \\
& \quad \quad \text{bindings}(\text{top}(\text{ctrl-stk})), \\
& \quad \quad \text{temp-stk})), \\
& \quad \text{translate-proc-list}(\text{proc-list}), \\
& \quad \text{list}(\text{list}('c-c, \\
& \quad \quad \text{mg-cond-to-p-nat}(\text{cc}(\text{mg-state}), \text{t-cond-list}))), \\
& \quad \text{MG-MAX-CTRL-STK-SIZE}, \\
& \quad \text{MG-MAX-TEMP-STK-SIZE}, \\
& \quad \text{MG-WORD-SIZE}, \\
& \quad 'run)) \\
= & \text{p-state}(\text{tag}('pc, \\
& \quad \text{cons}(\text{subr}, \\
& \quad \quad \text{if normal}(\text{mg-meaning-r}(\text{stmt}, \\
& \quad \quad \quad \text{proc-list}, \\
& \quad \quad \quad \text{mg-state}, \\
& \quad \quad \quad n, \\
& \quad \quad \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \quad \quad \text{p-ctrl-stk-size}(\text{ctrl-stk}))) \\
& \quad \quad \text{then length}(\text{code}(\text{translate}(\text{cinfo},
\end{aligned}$$

```

                                t-cond-list,
                                stmt,
                                proc-list)))
else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                proc-list,
                                                mg-state,
                                                n,
                                                list (length (temp-stk),
                                                            p-ctrl-stk-size (ctrl-stk))))),
                                label-alist (translate (cinfo,
                                                         t-cond-list,
                                                         stmt,
                                                         proc-list))),
                                append (code (translate (cinfo,
                                                         t-cond-list,
                                                         stmt,
                                                         proc-list)),
                                        code2)) endif)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (stmt,
                                          proc-list,
                                          mg-state,
                                          n,
                                          list (length (temp-stk),
                                                    p-ctrl-stk-size (ctrl-stk))))),
                bindings (top (ctrl-stk)),
                temp-stk),
translate-proc-list (proc-list),
list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-meaning-r (stmt,
                                                proc-list,
                                                mg-state,
                                                n,
                                                list (length (temp-stk),
                                                            p-ctrl-stk-size (ctrl-stk))))),
                                t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-index-array-exact-time-lemma

$((n \neq 0)$
 $\wedge \neg \text{resources-inadequatep}(\text{stmt},$

```

                                proc-list,
                                list (length (temp-stk),
                                        p-ctrl-stk-size (ctrl-stk))))
^ (car (stmt) = 'predefined-proc-call-mg)
^ (call-name (stmt) = 'mg-index-array)
^ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
^ ok-mg-def-plistp (proc-list)
^ ok-mg-statep (mg-state, r-cond-list)
^ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
              code2))
^ user-defined-procp (subr, proc-list)
^ listp (ctrl-stk)
^ all-cars-unique (mg-alist (mg-state))
^ signatures-match (mg-alist (mg-state), name-alist)
^ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                              bindings (top (ctrl-stk)),
                              temp-stk)
^ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
^ normal (mg-state))
→ (p (map-down (mg-state,
                proc-list,
                ctrl-stk,
                temp-stk,
                tag ('pc, cons (subr, length (code (cinfo))))),
                t-cond-list),
    clock (stmt, proc-list, mg-state, n))
= p-state (tag ('pc,
                cons (subr,
                    if normal (mg-meaning-r (stmt,
                                                proc-list,
                                                mg-state,
                                                n,
                                                list (length (temp-stk),
                                                        p-ctrl-stk-size (ctrl-stk))))
                    then length (code (translate (cinfo,
                                                    t-cond-list,
                                                    stmt,
                                                    proc-list)))
                    else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                                    proc-list,
                                                                    mg-state,
                                                                    n,
                                                                    list (length (temp-stk),

```

```

                                p-ctrl-stk-size (ctrl-stk))),
label-alist (translate (cinfo,
                        t-cond-list,
                        stmt,
                        proc-list)),
append (code (translate (cinfo,
                        t-cond-list,
                        stmt,
                        proc-list)),
code2)) endif)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (stmt,
proc-list,
mg-state,
n,
list (length (temp-stk),
p-ctrl-stk-size (ctrl-stk)))),
bindings (top (ctrl-stk)),
temp-stk),
translate-proc-list (proc-list),
list (list ('c-c,
mg-cond-to-p-nat (cc (mg-meaning-r (stmt,
proc-list,
mg-state,
n,
list (length (temp-stk),
p-ctrl-stk-size (ctrl-stk))))),
t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                 ;;
;;          EXACT-TIME-LEMMA MG-ARRAY-ELEMENT-ASSIGNMENT          ;;
;;                                                                 ;;
;;                                                                 ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

```

```
;; stmt looks like '(mg-predefined-proc-call mg-array-element-assignment (A i x size))
```

THEOREM: mg-array-element-assignment-args-have-simple-mg-type-refps
((car (stmt) = 'predefined-proc-call-mg)

```

 $\wedge$  (call-name (stmt) = 'mg-array-element-assignment)
 $\wedge$  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 $\wedge$  ok-mg-statep (mg-state, r-cond-list)
 $\wedge$  signatures-match (mg-alist (mg-state), name-alist)
 $\rightarrow$  (array-identifierp (car (call-actuals (stmt)), mg-alist (mg-state))
 $\wedge$  int-identifierp (cadr (call-actuals (stmt)), mg-alist (mg-state))
 $\wedge$  simple-typed-identifierp (caddr (call-actuals (stmt)),
array-elemtype (cadr (assoc (car (call-actuals (stmt)),
mg-alist (mg-state)))),
mg-alist (mg-state))
 $\wedge$  (caddr (call-actuals (stmt))
= array-length (cadr (assoc (car (call-actuals (stmt)),
mg-alist (mg-state))))))

```

THEOREM: mg-array-element-assignment-arg3-simple

```

((car (stmt) = 'predefined-proc-call-mg)
 $\wedge$  (call-name (stmt) = 'mg-array-element-assignment)
 $\wedge$  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 $\wedge$  ok-mg-statep (mg-state, r-cond-list)
 $\wedge$  signatures-match (mg-alist (mg-state), name-alist)
 $\rightarrow$  simple-identifierp (caddr (call-actuals (stmt)), mg-alist (mg-state))

```

THEOREM: mg-array-element-assignment-args-definedp

```

((car (stmt) = 'predefined-proc-call-mg)
 $\wedge$  (call-name (stmt) = 'mg-array-element-assignment)
 $\wedge$  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 $\wedge$  ok-mg-statep (mg-state, r-cond-list)
 $\wedge$  signatures-match (mg-alist (mg-state), name-alist)
 $\rightarrow$  (definedp (car (call-actuals (stmt)), mg-alist (mg-state))
 $\wedge$  definedp (cadr (call-actuals (stmt)), mg-alist (mg-state))
 $\wedge$  definedp (caddr (call-actuals (stmt)), mg-alist (mg-state)))

```

THEOREM: mg-array-element-assignment-arg4-small-integerp

```

((car (stmt) = 'predefined-proc-call-mg)
 $\wedge$  (call-name (stmt) = 'mg-array-element-assignment)
 $\wedge$  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 $\wedge$  ok-mg-statep (mg-state, r-cond-list)
 $\wedge$  signatures-match (mg-alist (mg-state), name-alist)
 $\rightarrow$  small-integerp (caddr (call-actuals (stmt)), 32)

```

THEOREM: not-zero-mg-array-element-assignment-arg4

```

((car (stmt) = 'predefined-proc-call-mg)
 $\wedge$  (call-name (stmt) = 'mg-array-element-assignment)
 $\wedge$  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 $\wedge$  ok-mg-statep (mg-state, r-cond-list)

```

$$\begin{aligned}
& \wedge \text{signatures-match}(\text{mg-alist}(\text{mg-state}), \text{name-alist}) \\
\rightarrow & ((\text{caddr}(\text{call-actuals}(\text{stmt})) \in \mathbf{N}) \\
& \wedge (\text{caddr}(\text{call-actuals}(\text{stmt})) \neq 0))
\end{aligned}$$

THEOREM: mg-array-element-assignment-steps-1-4

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge (\neg \text{resources-inadequate}(\text{stmt}, \\
& \quad \text{proc-list}, \\
& \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \text{p-ctrl-stk-size}(\text{ctrl-stk})))) \\
& \wedge (\text{car}(\text{stmt}) = \text{'predefined-proc-call-mg}) \\
& \wedge (\text{call-name}(\text{stmt}) = \text{'mg-array-element-assignment}) \\
& \wedge \text{ok-mg-statement}(\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}) \\
& \wedge \text{ok-mg-def-plistp}(\text{proc-list}) \\
& \wedge \text{ok-mg-statep}(\text{mg-state}, \text{r-cond-list}) \\
& \wedge (\text{code}(\text{translate-def-body}(\text{assoc}(\text{subr}, \text{proc-list}), \text{proc-list})) \\
& \quad = \text{append}(\text{code}(\text{translate}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list})), \\
& \quad \text{code2})) \\
& \wedge \text{user-defined-procp}(\text{subr}, \text{proc-list}) \\
& \wedge \text{listp}(\text{ctrl-stk}) \\
& \wedge \text{all-cars-unique}(\text{mg-alist}(\text{mg-state})) \\
& \wedge \text{signatures-match}(\text{mg-alist}(\text{mg-state}), \text{name-alist}) \\
& \wedge \text{mg-vars-list-ok-in-p-state}(\text{mg-alist}(\text{mg-state}), \\
& \quad \text{bindings}(\text{top}(\text{ctrl-stk})), \\
& \quad \text{temp-stk}) \\
& \wedge \text{no-p-aliasing}(\text{bindings}(\text{top}(\text{ctrl-stk})), \text{mg-alist}(\text{mg-state})) \\
& \wedge \text{normal}(\text{mg-state})) \\
\rightarrow & (\text{p-step}(\text{p-step}(\text{p-step}(\text{p-step}(\text{map-down}(\text{mg-state}, \\
& \quad \text{proc-list}, \\
& \quad \text{ctrl-stk}, \\
& \quad \text{temp-stk}, \\
& \quad \text{tag}(\text{'pc}, \\
& \quad \quad \text{cons}(\text{subr}, \\
& \quad \quad \quad \text{length}(\text{code}(\text{cinfo})))), \\
& \quad \quad \text{t-cond-list})))))) \\
& = \text{p-state}(\text{tag}(\text{'pc}, \text{cons}(\text{subr}, \text{length}(\text{code}(\text{cinfo})) + 4)), \\
& \quad \text{ctrl-stk}, \\
& \quad \text{push}(\text{tag}(\text{'int}, \text{caddr}(\text{call-actuals}(\text{stmt}))), \\
& \quad \quad \text{push}(\text{value}(\text{caddr}(\text{call-actuals}(\text{stmt})), \\
& \quad \quad \quad \text{bindings}(\text{top}(\text{ctrl-stk}))), \\
& \quad \quad \text{push}(\text{value}(\text{cadr}(\text{call-actuals}(\text{stmt})), \\
& \quad \quad \quad \text{bindings}(\text{top}(\text{ctrl-stk}))), \\
& \quad \quad \text{push}(\text{value}(\text{car}(\text{call-actuals}(\text{stmt})), \\
& \quad \quad \quad \text{bindings}(\text{top}(\text{ctrl-stk}))),
\end{aligned}$$

```

map-down-values (mg-alist (mg-state),
                   bindings (top (ctrl-stk)),
                   temp-stk))))),
translate-proc-list (proc-list),
list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-array-element-assignment-step-5

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
                             proc-list,
                             list (length (temp-stk),
                                   p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-array-element-assignment)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp (proc-list)
 ∧ ok-mg-statep (mg-state, r-cond-list)
 ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
              code2))
 ∧ user-defined-procp (subr, proc-list)
 ∧ listp (ctrl-stk)
 ∧ all-cars-unique (mg-alist (mg-state))
 ∧ signatures-match (mg-alist (mg-state), name-alist)
 ∧ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                               bindings (top (ctrl-stk)),
                               temp-stk)
 ∧ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 ∧ normal (mg-state))
→ (p-step (p-state (tag ('pc, cons (subr, length (code (cinfo)) + 4)),
                        ctrl-stk,
                        push (tag ('int, caddr (call-actuals (stmt))),
                            push (value (caddr (call-actuals (stmt)),
                                                bindings (top (ctrl-stk))),
                                push (value (cadr (call-actuals (stmt)),
                                                bindings (top (ctrl-stk))),
                                    push (value (car (call-actuals (stmt)),
                                                bindings (top (ctrl-stk))),
                                        map-down-values (mg-alist (mg-state),

```

```

bindings (top (ctrl-stk)),
temp-stk))))),

translate-proc-list (proc-list),
list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
= p-state (tag ('pc,
               '(mg-array-element-assignment . 0)),
  push (p-frame (cons (cons ('a,
                             value (car (call-actuals (stmt)),
                             bindings (top (ctrl-stk)))))
                     cons (cons ('i,
                             value (cadr (call-actuals (stmt)),
                             bindings (top (ctrl-stk)))))
                     cons (cons ('value,
                             value (caddr (call-actuals (stmt)),
                             bindings (top (ctrl-stk)))))
                     cons (cons ('array-size,
                             tag ('int,
                             caddr (call-actuals (stmt)))))
                     '((temp-i nat 0)))))),
    tag ('pc,
        cons (subr, length (code (cinfo))
              + 5))),
  ctrl-stk),
map-down-values (mg-alist (mg-state),
                bindings (top (ctrl-stk)),
                temp-stk),
translate-proc-list (proc-list),
list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-array-element-assignment-steps-6-8

$((n \neq 0)$
 $\wedge \quad (\neg \text{resources-inadequatep} (stmt,$
 $\quad \quad \quad proc-list,$
 $\quad \quad \quad list (length (temp-stk),$

```

                                p-ctrl-stk-size (ctrl-stk))))
^ (car (stmt) = 'predefined-proc-call-mg)
^ (call-name (stmt) = 'mg-array-element-assignment)
^ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
^ ok-mg-def-plistp (proc-list)
^ ok-mg-statep (mg-state, r-cond-list)
^ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
              code2))
^ user-defined-procp (subr, proc-list)
^ listp (ctrl-stk)
^ all-cars-unique (mg-alist (mg-state))
^ signatures-match (mg-alist (mg-state), name-alist)
^ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                              bindings (top (ctrl-stk)),
                              temp-stk)
^ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
^ normal (mg-state))
→ (p-step (p-step (p-step (p-state (tag ('pc,
                                         '(mg-array-element-assignment
                                           . 0)),
                                         push (p-frame (cons (cons ('a,
                                                                    value (car (call-actuals (stmt)),
                                                                    bindings (top (ctrl-stk)))),
                                                                    cons (cons ('i,
                                                                    value (cadr (call-actuals (stmt)),
                                                                    bindings (top (ctrl-stk)))),
                                                                    cons (cons ('value,
                                                                    value (caddr (call-actuals (stmt)),
                                                                    bindings (top (ctrl-stk)))),
                                                                    cons (cons ('array-size,
                                                                    tag ('int,
                                                                    caddr (call-actuals (stmt)
                                                                    '((temp-i
                                                                    nat
                                                                    0)))))),
                                                                    tag ('pc,
                                                                    cons (subr,
                                                                    length (code (cinfo))
                                                                    + 5))),
                                         ctrl-stk),
                                         map-down-values (mg-alist (mg-state),
                                                             bindings (top (ctrl-stk)),
                                                             temp-stk),

```

```

translate-proc-list (proc-list),
list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-state),
                                t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))))
= p-state (tag ('pc,
               '(mg-array-element-assignment . 3)),
  push (p-frame (list (cons ('a,
                             value (car (call-actuals (stmt)),
                             bindings (top (ctrl-stk)))),
                    cons ('i,
                          value (cadr (call-actuals (stmt)),
                          bindings (top (ctrl-stk)))),
                    cons ('value,
                          value (caddr (call-actuals (stmt)),
                          bindings (top (ctrl-stk)))),
                    cons ('array-size,
                          tag ('int,
                              caddr (call-actuals (stmt)))),
                    cons ('temp-i,
                          mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                mg-alist (mg-state)))))),
          tag ('pc,
              cons (subr, length (code (cinfo)
                                         + 5))),
          ctrl-stk),
  push (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                              mg-alist (mg-state)))),
    map-down-values (mg-alist (mg-state),
                          bindings (top (ctrl-stk)),
                          temp-stk)),
  translate-proc-list (proc-list),
  list (list ('c-c,
             mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
  MG-MAX-CTRL-STK-SIZE,
  MG-MAX-TEMP-STK-SIZE,
  MG-WORD-SIZE,
  'run))

```

THEOREM: mg-array-element-assignment-steps-9-12-neg-index
 $((n \neq 0)$

```

 $\wedge$  ( $\neg$  resources-inadequatep (stmt,
                               proc-list,
                               list (length (temp-stk),
                                       p-ctrl-stk-size (ctrl-stk))))
 $\wedge$  (car (stmt) = 'predefined-proc-call-mg)
 $\wedge$  (call-name (stmt) = 'mg-array-element-assignment)
 $\wedge$  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 $\wedge$  ok-mg-def-plistp (proc-list)
 $\wedge$  ok-mg-statep (mg-state, r-cond-list)
 $\wedge$  (code (translate-def-body (assoc (subr, proc-list), proc-list))
      = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                code2))
 $\wedge$  user-defined-procp (subr, proc-list)
 $\wedge$  listp (ctrl-stk)
 $\wedge$  all-cars-unique (mg-alist (mg-state))
 $\wedge$  signatures-match (mg-alist (mg-state), name-alist)
 $\wedge$  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                         bindings (top (ctrl-stk)),
                                         temp-stk)
 $\wedge$  no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 $\wedge$  normal (mg-state)
 $\wedge$  negativep (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                            mg-alist (mg-state)))))))
 $\rightarrow$  (p-step (p-step (p-step (p-step (p-state (tag ('pc,
                                                    '(mg-array-element-assignment
                                                      . 3)),
                                                    push (p-frame (list (cons ('a,
                                                                              value (car (call-actuals (stmt)),
                                                                              bindings (top (ctrl-stk))),
                                                                              cons ('i,
                                                                              value (cadr (call-actuals (stmt)),
                                                                              bindings (top (ctrl-stk))),
                                                                              cons ('value,
                                                                              value (caddr (call-actuals (stmt)),
                                                                              bindings (top (ctrl-stk))),
                                                                              cons ('array-size,
                                                                              tag ('int,
                                                                              caddr (call-actuals (stmt))),
                                                                              cons ('temp-i,
                                                                              mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                              mg-alist (mg-state)))))),
                                                                              tag ('pc,
                                                                              cons (subr,
                                                                              length (code (cinfo))
                                                                              mg-e

```

$$\begin{aligned}
& \quad \quad \quad + \ 5))), \\
& \quad \quad \quad ctrl-stk), \\
& \quad \quad \quad \text{push}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{cadr}(\text{call-actuals}(stmt), \\
& \quad \quad \quad \quad \quad \quad \text{mg-alist}(mg-state)))), \\
& \quad \quad \quad \text{map-down-values}(\text{mg-alist}(mg-state), \\
& \quad \quad \quad \quad \quad \quad \text{bindings}(\text{top}(ctrl-stk)), \\
& \quad \quad \quad \quad \quad \quad \text{temp-stk})), \\
& \quad \quad \quad \text{translate-proc-list}(proc-list), \\
& \quad \quad \quad \text{list}(\text{list}('c-c, \\
& \quad \quad \quad \quad \quad \quad \text{mg-cond-to-p-nat}(\text{cc}(mg-state), \\
& \quad \quad \quad \quad \quad \quad \quad \quad \text{t-cond-list}))), \\
& \quad \quad \quad \text{MG-MAX-CTRL-STK-SIZE}, \\
& \quad \quad \quad \text{MG-MAX-TEMP-STK-SIZE}, \\
& \quad \quad \quad \text{MG-WORD-SIZE}, \\
& \quad \quad \quad 'run)))))) \\
= & \quad \text{p-state}(\text{tag}('pc, \text{cons}(subr, \text{length}(\text{code}(cinfo)) + 5)), \\
& \quad \quad \quad ctrl-stk, \\
& \quad \quad \quad \text{map-down-values}(\text{mg-alist}(mg-state), \\
& \quad \quad \quad \quad \quad \quad \text{bindings}(\text{top}(ctrl-stk)), \\
& \quad \quad \quad \quad \quad \quad \text{temp-stk}), \\
& \quad \quad \quad \text{translate-proc-list}(proc-list), \\
& \quad \quad \quad \text{list}(\text{list}('c-c, '(\text{nat } 1))), \\
& \quad \quad \quad \text{MG-MAX-CTRL-STK-SIZE}, \\
& \quad \quad \quad \text{MG-MAX-TEMP-STK-SIZE}, \\
& \quad \quad \quad \text{MG-WORD-SIZE}, \\
& \quad \quad \quad 'run))
\end{aligned}$$

THEOREM: mg-array-element-assignment-steps-9-11-no-error

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge \quad (\neg \text{resources-inadequatep}(stmt, \\
& \quad \quad \quad \text{proc-list}, \\
& \quad \quad \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \quad \quad \quad \quad \text{p-ctrl-stk-size}(ctrl-stk)))) \\
& \wedge \quad (\text{car}(stmt) = 'predefined-proc-call-mg) \\
& \wedge \quad (\text{call-name}(stmt) = 'mg-array-element-assignment) \\
& \wedge \quad \text{ok-mg-statement}(stmt, r-cond-list, name-alist, proc-list) \\
& \wedge \quad \text{ok-mg-def-plistp}(proc-list) \\
& \wedge \quad \text{ok-mg-statep}(mg-state, r-cond-list) \\
& \wedge \quad (\text{code}(\text{translate-def-body}(\text{assoc}(subr, proc-list), proc-list)) \\
& \quad = \text{append}(\text{code}(\text{translate}(cinfo, t-cond-list, stmt, proc-list)), \\
& \quad \quad \quad \text{code2})) \\
& \wedge \quad \text{user-defined-procp}(subr, proc-list) \\
& \wedge \quad \text{listp}(ctrl-stk) \\
& \wedge \quad \text{all-cars-unique}(\text{mg-alist}(mg-state))
\end{aligned}$$

```

^ signatures-match (mg-alist (mg-state), name-alist)
^ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                               bindings (top (ctrl-stk)),
                               temp-stk)
^ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
^ normal (mg-state)
^ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                            mg-alist (mg-state))))))))
→ (p-step (p-step (p-step (p-state (tag ('pc,
                                         '(mg-array-element-assignment
                                           . 3)),
                                         push (p-frame (list (cons ('a,
                                                                    value (car (call-actuals (stmt)),
                                                                    bindings (top (ctrl-stk)))),
                                                                    cons ('i,
                                                                    value (cadr (call-actuals (stmt)),
                                                                    bindings (top (ctrl-stk)))),
                                                                    cons ('value,
                                                                    value (caddr (call-actuals (stmt)),
                                                                    bindings (top (ctrl-stk)))),
                                                                    cons ('array-size,
                                                                    tag ('int,
                                                                    caddr (call-actuals (stmt)))),
                                                                    cons ('temp-i,
                                                                    mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                    mg-alist (mg-state))))),
                                                                    tag ('pc,
                                                                    cons (subr,
                                                                    length (code (cinfo))
                                                                    + 5))),
                                         ctrl-stk),
                                         push (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                    mg-alist (mg-state))))),
                                         map-down-values (mg-alist (mg-state),
                                                            bindings (top (ctrl-stk)),
                                                            temp-stk)),
                                         translate-proc-list (proc-list),
                                         list (list ('c-c,
                                                       mg-cond-to-p-nat (cc (mg-state),
                                                       t-cond-list))),
                                         MG-MAX-CTRL-STK-SIZE,
                                         MG-MAX-TEMP-STK-SIZE,
                                         MG-WORD-SIZE,
                                         'run))))))

```

```

=  p-state (tag ('pc,
                '(mg-array-element-assignment . 6)),
      push (p-frame (list (cons ('a,
                                value (car (call-actuals (stmt)),
                                bindings (top (ctrl-stk)))),
                          cons ('i,
                                value (cadr (call-actuals (stmt)),
                                bindings (top (ctrl-stk)))),
                          cons ('value,
                                value (caddr (call-actuals (stmt)),
                                bindings (top (ctrl-stk)))),
                          cons ('array-size,
                                tag ('int,
                                    caddr (call-actuals (stmt)))),
                          cons ('temp-i,
                                mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                    mg-alist (mg-state)))))),
                                tag ('pc,
                                    cons (subr, length (code (cinfo))
                                          + 5))),
          ctrl-stk),
      push (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                  mg-alist (mg-state)))),
          push (tag ('int, caddr (call-actuals (stmt))),
              map-down-values (mg-alist (mg-state),
                                bindings (top (ctrl-stk)),
                                temp-stk))),
      translate-proc-list (proc-list),
      list (list ('c-c,
                  mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
      MG-MAX-CTRL-STK-SIZE,
      MG-MAX-TEMP-STK-SIZE,
      MG-WORD-SIZE,
      'run))

```

THEOREM: mg-array-element-assignment-step-12-no-error

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
                             proc-list,
                             list (length (temp-stk),
                                     p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-array-element-assignment)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)

```

```

^ ok-mg-def-plistp (proc-list)
^ ok-mg-statep (mg-state, r-cond-list)
^ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
              code2))
^ user-defined-procp (subr, proc-list)
^ listp (ctrl-stk)
^ all-cars-unique (mg-alist (mg-state))
^ signatures-match (mg-alist (mg-state), name-alist)
^ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                     bindings (top (ctrl-stk)),
                                     temp-stk)
^ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
^ normal (mg-state)
^ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                             mg-alist (mg-state))))))))
→ (p-step (p-state (tag ('pc,
                        '(mg-array-element-assignment . 6)),
            push (p-frame (list (cons ('a,
                                      value (car (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))),
                                cons ('i,
                                      value (cadr (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))),
                                cons ('value,
                                      value (caddr (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))),
                                cons ('array-size,
                                      tag ('int,
                                      caddr (call-actuals (stmt))),
                                cons ('temp-i,
                                      mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                              mg-alist (mg-state)))))),
                                tag ('pc,
                                cons (subr, length (code (cinfo))
                                + 5))),
                    ctrl-stk),
            push (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                         mg-alist (mg-state)))),
            push (tag ('int, caddr (call-actuals (stmt)),
                    map-down-values (mg-alist (mg-state),
                    bindings (top (ctrl-stk)),
                    temp-stk))),
            translate-proc-list (proc-list),

```

```

list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
= p-state (tag ('pc,
               '(mg-array-element-assignment . 7)),
           push (p-frame (list (cons ('a,
                                     value (car (call-actuals (stmt)),
                                     bindings (top (ctrl-stk)))),
                                cons ('i,
                                     value (cadr (call-actuals (stmt)),
                                     bindings (top (ctrl-stk)))),
                                cons ('value,
                                     value (caddr (call-actuals (stmt)),
                                     bindings (top (ctrl-stk)))),
                                cons ('array-size,
                                     tag ('int,
                                     caddr (call-actuals (stmt)))),
                                cons ('temp-i,
                                     mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                     mg-alist (mg-state))))))),
               tag ('pc,
                   cons (subr, length (code (cinfo))
                       + 5))),
           ctrl-stk),
push (tag ('int,
          idifference (caddr (call-actuals (stmt)),
                      untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                      mg-alist (mg-state))))))),
      map-down-values (mg-alist (mg-state),
                      bindings (top (ctrl-stk)),
                      temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-array-element-assignment-step-13-index-error
 $((n \neq 0)$

$$\begin{aligned}
& \wedge (\neg \text{resources-inadequatep} (stmt, \\
& \quad \text{proc-list}, \\
& \quad \text{list} (\text{length} (temp-stk), \\
& \quad \quad \text{p-ctrl-stk-size} (ctrl-stk)))) \\
& \wedge (\text{car} (stmt) = \text{'predefined-proc-call-mg}) \\
& \wedge (\text{call-name} (stmt) = \text{'mg-array-element-assignment}) \\
& \wedge \text{ok-mg-statement} (stmt, r-cond-list, name-alist, proc-list) \\
& \wedge \text{ok-mg-def-plistp} (proc-list) \\
& \wedge \text{ok-mg-statep} (mg-state, r-cond-list) \\
& \wedge (\text{code} (\text{translate-def-body} (\text{assoc} (subr, proc-list), proc-list)) \\
& \quad = \text{append} (\text{code} (\text{translate} (cinfo, t-cond-list, stmt, proc-list)), \\
& \quad \quad \text{code2})) \\
& \wedge \text{user-defined-procp} (subr, proc-list) \\
& \wedge \text{listp} (ctrl-stk) \\
& \wedge \text{all-cars-unique} (\text{mg-alist} (mg-state)) \\
& \wedge \text{signatures-match} (\text{mg-alist} (mg-state), name-alist) \\
& \wedge \text{mg-vars-list-ok-in-p-state} (\text{mg-alist} (mg-state), \\
& \quad \text{bindings} (\text{top} (ctrl-stk)), \\
& \quad \text{temp-stk}) \\
& \wedge \text{no-p-aliasing} (\text{bindings} (\text{top} (ctrl-stk)), \text{mg-alist} (mg-state)) \\
& \wedge \text{normal} (mg-state) \\
& \wedge (\neg \text{negativep} (\text{untag} (\text{mg-to-p-simple-literal} (\text{caddr} (\text{assoc} (\text{cadr} (\text{call-actuals} (stmt)), \\
& \quad \text{mg-alist} (mg-state))))))) \\
& \wedge (\text{idifference} (\text{caddr} (\text{call-actuals} (stmt)), \\
& \quad \text{untag} (\text{mg-to-p-simple-literal} (\text{caddr} (\text{assoc} (\text{cadr} (\text{call-actuals} (stmt)), \\
& \quad \text{mg-alist} (mg-state))))))) \simeq 0)) \\
\rightarrow & (\text{p-step} (\text{p-state} (\text{tag} ('pc, \\
& \quad \text{'(mg-array-element-assignment . 7)}), \\
& \quad \text{push} (\text{p-frame} (\text{list} (\text{cons} ('a, \\
& \quad \quad \text{value} (\text{car} (\text{call-actuals} (stmt)), \\
& \quad \quad \text{bindings} (\text{top} (ctrl-stk)))), \\
& \quad \text{cons} ('i, \\
& \quad \quad \text{value} (\text{cadr} (\text{call-actuals} (stmt)), \\
& \quad \quad \text{bindings} (\text{top} (ctrl-stk)))), \\
& \quad \text{cons} ('value, \\
& \quad \quad \text{value} (\text{caddr} (\text{call-actuals} (stmt)), \\
& \quad \quad \text{bindings} (\text{top} (ctrl-stk)))), \\
& \quad \text{cons} ('array-size, \\
& \quad \quad \text{tag} ('int, \\
& \quad \quad \text{caddr} (\text{call-actuals} (stmt)))), \\
& \quad \text{cons} ('temp-i, \\
& \quad \quad \text{mg-to-p-simple-literal} (\text{caddr} (\text{assoc} (\text{cadr} (\text{call-actuals} (stmt)), \\
& \quad \quad \text{mg-alist} (mg-state))))))), \\
& \quad \text{tag} ('pc,
\end{aligned}$$

```

cons (subr, length (code (cinfo))
      + 5)),
ctrl-stk),
push (tag ('int,
          idifference (caddr (call-actuals (stmt)),
                      untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt))
                                                                    mg-alist (mg-state)))))),
          map-down-values (mg-alist (mg-state),
                              bindings (top (ctrl-stk)),
                              temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
= p-state (tag ('pc,
               '(mg-array-element-assignment . 16)),
           push (p-frame (list (cons ('a,
                                      value (car (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))),
                                cons ('i,
                                      value (cadr (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))),
                                cons ('value,
                                      value (caddr (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))),
                                cons ('array-size,
                                      tag ('int,
                                          caddr (call-actuals (stmt)))),
                                cons ('temp-i,
                                      mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt))
                                                                    mg-alist (mg-state))))))),
               tag ('pc,
                   cons (subr, length (code (cinfo))
                         + 5)),
               ctrl-stk),
map-down-values (mg-alist (mg-state),
                bindings (top (ctrl-stk)),
                temp-stk),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),

```

MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: mg-array-element-assignment-steps-14-16-index-error

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge (\neg \text{resources-inadequatep} (stmt, \\
& \quad \quad \quad \text{proc-list}, \\
& \quad \quad \quad \text{list (length (temp-stk),} \\
& \quad \quad \quad \text{p-ctrl-stk-size (ctrl-stk))})) \\
& \wedge (\text{car (stmt)} = \text{'predefined-proc-call-mg}) \\
& \wedge (\text{call-name (stmt)} = \text{'mg-array-element-assignment}) \\
& \wedge \text{ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)} \\
& \wedge \text{ok-mg-def-plistp (proc-list)} \\
& \wedge \text{ok-mg-statep (mg-state, r-cond-list)} \\
& \wedge (\text{code (translate-def-body (assoc (subr, proc-list), proc-list))} \\
& \quad = \text{append (code (translate (cinfo, t-cond-list, stmt, proc-list)),} \\
& \quad \quad \text{code2})) \\
& \wedge \text{user-defined-procp (subr, proc-list)} \\
& \wedge \text{listp (ctrl-stk)} \\
& \wedge \text{all-cars-unique (mg-alist (mg-state))} \\
& \wedge \text{signatures-match (mg-alist (mg-state), name-alist)} \\
& \wedge \text{mg-vars-list-ok-in-p-state (mg-alist (mg-state),} \\
& \quad \quad \text{bindings (top (ctrl-stk)),} \\
& \quad \quad \text{temp-stk)} \\
& \wedge \text{no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))} \\
& \wedge \text{normal (mg-state)} \\
& \wedge (\neg \text{negativep (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),} \\
& \quad \quad \quad \text{mg-alist (mg-state)))))))} \\
& \wedge (\text{idifference (caddr (call-actuals (stmt)),} \\
& \quad \quad \text{untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),} \\
& \quad \quad \quad \text{mg-alist (mg-state))))))} \simeq 0)) \\
& \rightarrow (\text{p-step (p-step (p-step (p-state (tag ('pc,} \\
& \quad \quad \quad \text{'(mg-array-element-assignment} \\
& \quad \quad \quad \text{. 16)),} \\
& \quad \quad \text{push (p-frame (list (cons ('a,} \\
& \quad \quad \quad \text{value (car (call-actuals (stmt)),} \\
& \quad \quad \quad \text{bindings (top (ctrl-stk)))),} \\
& \quad \quad \text{cons ('i,} \\
& \quad \quad \quad \text{value (cadr (call-actuals (stmt)),} \\
& \quad \quad \quad \text{bindings (top (ctrl-stk)))),} \\
& \quad \quad \text{cons ('value,} \\
& \quad \quad \quad \text{value (caddr (call-actuals (stmt)),}
\end{aligned}$$

$$\begin{aligned}
& \text{bindings}(\text{top}(\text{ctrl-stk}))), \\
& \text{cons}(\text{'array-size}, \\
& \quad \text{tag}(\text{'int}, \\
& \quad \quad \text{caddr}(\text{call-actuals}(\text{stmt}))), \\
& \text{cons}(\text{'temp-i}, \\
& \quad \text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{cadr}(\text{call-actuals}(\text{stmt})), \\
& \quad \quad \text{mg-alist}(\text{mg-state}))), \\
& \text{tag}(\text{'pc}, \\
& \quad \text{cons}(\text{subr}, \\
& \quad \quad \text{length}(\text{code}(\text{cinfo})) \\
& \quad \quad + 5))), \\
& \text{ctrl-stk}), \\
& \text{map-down-values}(\text{mg-alist}(\text{mg-state}), \\
& \quad \text{bindings}(\text{top}(\text{ctrl-stk})), \\
& \quad \text{temp-stk}), \\
& \text{translate-proc-list}(\text{proc-list}), \\
& \text{list}(\text{list}(\text{'c-c}, \\
& \quad \text{mg-cond-to-p-nat}(\text{cc}(\text{mg-state}), \\
& \quad \quad \text{t-cond-list}))), \\
& \text{MG-MAX-CTRL-STK-SIZE}, \\
& \text{MG-MAX-TEMP-STK-SIZE}, \\
& \text{MG-WORD-SIZE}, \\
& \text{'run}))) \\
= & \text{p-state}(\text{tag}(\text{'pc}, \text{cons}(\text{subr}, \text{length}(\text{code}(\text{cinfo})) + 5)), \\
& \text{ctrl-stk}, \\
& \text{map-down-values}(\text{mg-alist}(\text{mg-state}), \\
& \quad \text{bindings}(\text{top}(\text{ctrl-stk})), \\
& \quad \text{temp-stk}), \\
& \text{translate-proc-list}(\text{proc-list}), \\
& \text{list}(\text{list}(\text{'c-c}, \text{'(nat 1)})), \\
& \text{MG-MAX-CTRL-STK-SIZE}, \\
& \text{MG-MAX-TEMP-STK-SIZE}, \\
& \text{MG-WORD-SIZE}, \\
& \text{'run}))
\end{aligned}$$

;; Need 13-15 and 16-17 no-error lemmas

THEOREM: mg-array-element-assignment-step-13-no-error

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge (\neg \text{resources-inadequatep}(\text{stmt}, \\
& \quad \text{proc-list}, \\
& \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \quad \text{p-ctrl-stk-size}(\text{ctrl-stk}))))
\end{aligned}$$

```

^ (car (stmt) = 'predefined-proc-call-mg)
^ (call-name (stmt) = 'mg-array-element-assignment)
^ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
^ ok-mg-def-plistp (proc-list)
^ ok-mg-statep (mg-state, r-cond-list)
^ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
              code2))
^ user-defined-procp (subr, proc-list)
^ listp (ctrl-stk)
^ all-cars-unique (mg-alist (mg-state))
^ signatures-match (mg-alist (mg-state), name-alist)
^ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                              bindings (top (ctrl-stk)),
                              temp-stk)
^ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
^ normal (mg-state)
^ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                            mg-alist (mg-state)))))))
^ (idifference (caddr (call-actuals (stmt)),
               untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                            mg-alist (mg-state))))))) ≠ 0)
→ (p-step (p-state (tag ('pc,
                        '(mg-array-element-assignment . 7)),
            push (p-frame (list (cons ('a,
                                      value (car (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))),
                                cons ('i,
                                      value (cadr (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))),
                                cons ('value,
                                      value (caddr (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))),
                                cons ('array-size,
                                      tag ('int,
                                      caddr (call-actuals (stmt)))),
                                cons ('temp-i,
                                      mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                              mg-alist (mg-state))))))),
            tag ('pc,
                cons (subr, length (code (cinfo)
                                      + 5))),
            ctrl-stk),
    push (tag ('int,

```

```

                                idifference (caddr (call-actuals (stmt)),
                                                untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)
                                                                                               mg-alist (mg-state))))))),
                                map-down-values (mg-alist (mg-state),
                                                            bindings (top (ctrl-stk)),
                                                            temp-stk)),
                                translate-proc-list (proc-list),
                                list (list ('c-c,
                                             mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                                MG-MAX-CTRL-STK-SIZE,
                                MG-MAX-TEMP-STK-SIZE,
                                MG-WORD-SIZE,
                                'run))
=  p-state (tag ('pc,
                 '(mg-array-element-assignment . 8)),
        push (p-frame (list (cons ('a,
                                    value (car (call-actuals (stmt)),
                                    bindings (top (ctrl-stk)))),
                                cons ('i,
                                    value (cadr (call-actuals (stmt)),
                                    bindings (top (ctrl-stk)))),
                                cons ('value,
                                    value (caddr (call-actuals (stmt)),
                                    bindings (top (ctrl-stk)))),
                                cons ('array-size,
                                    tag ('int,
                                        caddr (call-actuals (stmt)))),
                                cons ('temp-i,
                                    mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                                               mg-alist (mg-state))))))),
                                tag ('pc,
                                    cons (subr, length (code (cinfo)
                                                                + 5))),
                                ctrl-stk),
        map-down-values (mg-alist (mg-state),
                            bindings (top (ctrl-stk)),
                            temp-stk),
        translate-proc-list (proc-list),
        list (list ('c-c,
                     mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
        MG-MAX-CTRL-STK-SIZE,
        MG-MAX-TEMP-STK-SIZE,
        MG-WORD-SIZE,
        'run))

```

THEOREM: mg-array-element-assignment-step-14-no-error

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge (\neg \text{resources-inadequatep} (stmt, \\
& \quad \text{proc-list}, \\
& \quad \text{list (length (temp-stk),} \\
& \quad \quad \text{p-ctrl-stk-size (ctrl-stk))})) \\
& \wedge (\text{car (stmt) = 'predefined-proc-call-mg}) \\
& \wedge (\text{call-name (stmt) = 'mg-array-element-assignment}) \\
& \wedge \text{ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)} \\
& \wedge \text{ok-mg-def-plistp (proc-list)} \\
& \wedge \text{ok-mg-statep (mg-state, r-cond-list)} \\
& \wedge (\text{code (translate-def-body (assoc (subr, proc-list), proc-list))} \\
& \quad = \text{append (code (translate (cinfo, t-cond-list, stmt, proc-list)),} \\
& \quad \quad \text{code2})) \\
& \wedge \text{user-defined-procp (subr, proc-list)} \\
& \wedge \text{listp (ctrl-stk)} \\
& \wedge \text{all-cars-unique (mg-alist (mg-state))} \\
& \wedge \text{signatures-match (mg-alist (mg-state), name-alist)} \\
& \wedge \text{mg-vars-list-ok-in-p-state (mg-alist (mg-state),} \\
& \quad \text{bindings (top (ctrl-stk)),} \\
& \quad \text{temp-stk)} \\
& \wedge \text{no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))} \\
& \wedge \text{normal (mg-state)} \\
& \wedge (\neg \text{negativep (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),} \\
& \quad \quad \text{mg-alist (mg-state)))))))} \\
& \wedge (\text{idifference (caddr (call-actuals (stmt)),} \\
& \quad \text{untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),} \\
& \quad \quad \text{mg-alist (mg-state))))))} \neq 0)) \\
& \rightarrow (\text{p-step (p-state (tag ('pc,} \\
& \quad \text{'(mg-array-element-assignment . 8)),} \\
& \quad \text{push (p-frame (list (cons ('a,} \\
& \quad \quad \text{value (car (call-actuals (stmt)),} \\
& \quad \quad \text{bindings (top (ctrl-stk))}),} \\
& \quad \text{cons ('i,} \\
& \quad \quad \text{value (cadr (call-actuals (stmt)),} \\
& \quad \quad \text{bindings (top (ctrl-stk))}),} \\
& \quad \text{cons ('value,} \\
& \quad \quad \text{value (caddr (call-actuals (stmt)),} \\
& \quad \quad \text{bindings (top (ctrl-stk))}),} \\
& \quad \text{cons ('array-size,} \\
& \quad \quad \text{tag ('int,} \\
& \quad \quad \text{caddr (call-actuals (stmt))}),} \\
& \quad \text{cons ('temp-i,} \\
& \quad \quad \text{mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),}
\end{aligned}$$

```

tag('pc,
    cons(subr, length(code(cinfo))
        + 5)),
ctrl-stk),
map-down-values(mg-alist(mg-state),
    bindings(top(ctrl-stk)),
    temp-stk),
translate-proc-list(proc-list),
list(list('c-c,
    mg-cond-to-p-nat(cc(mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
= p-state(tag('pc,
    '(mg-array-element-assignment . 9)),
    push(p-frame(list(cons('a,
        value(car(call-actuals(stmt)),
        bindings(top(ctrl-stk)))),
        cons('i,
            value(cadr(call-actuals(stmt)),
            bindings(top(ctrl-stk)))),
        cons('value,
            value(caddr(call-actuals(stmt)),
            bindings(top(ctrl-stk)))),
        cons('array-size,
            tag('int,
                caddr(call-actuals(stmt)))),
        cons('temp-i,
            mg-to-p-simple-literal(caddr(assoc(cadr(call-actuals(stmt)),
                mg-alist(mg-state)))))),
        tag('pc,
            cons(subr, length(code(cinfo))
                + 5)),
        ctrl-stk),
        push(value('value,
            list(cons('a,
                value(car(call-actuals(stmt)),
                bindings(top(ctrl-stk)))),
                cons('i,
                    value(cadr(call-actuals(stmt)),
                    bindings(top(ctrl-stk)))),
                cons('value,

```

```

value (caddr (call-actuals (stmt)),
        bindings (top (ctrl-stk))),
cons ('array-size,
      tag ('int,
            caddr (call-actuals (stmt))),
cons ('temp-i,
      mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                             mg-alist (mg-state)))))),
map-down-values (mg-alist (mg-state),
                    bindings (top (ctrl-stk)),
                    temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-array-element-assignment-step-15-no-error

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
                             proc-list,
                             list (length (temp-stk),
                                       p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-array-element-assignment)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp (proc-list)
 ∧ ok-mg-statep (mg-state, r-cond-list)
 ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
              code2))
 ∧ user-defined-procp (subr, proc-list)
 ∧ listp (ctrl-stk)
 ∧ all-cars-unique (mg-alist (mg-state))
 ∧ signatures-match (mg-alist (mg-state), name-alist)
 ∧ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                       bindings (top (ctrl-stk)),
                                       temp-stk)
 ∧ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 ∧ normal (mg-state)
 ∧ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                             mg-alist (mg-state)))))))

```

$$\begin{aligned}
& \wedge \quad (\text{idifference} (\text{caddr} (\text{call-actuals} (stmt)), \\
& \quad \text{untag} (\text{mg-to-p-simple-literal} (\text{caddr} (\text{assoc} (\text{cadr} (\text{call-actuals} (stmt)), \\
& \quad \quad \text{mg-alist} (mg-state)))))) \neq 0)) \\
\rightarrow & \quad (\text{p-step} (\text{p-state} (\text{tag} ('pc, \\
& \quad \quad \text{'(mg-array-element-assignment . 9)}), \\
& \quad \text{push} (\text{p-frame} (\text{list} (\text{cons} ('a, \\
& \quad \quad \text{value} (\text{car} (\text{call-actuals} (stmt)), \\
& \quad \quad \text{bindings} (\text{top} (ctrl-stk)))), \\
& \quad \text{cons} ('i, \\
& \quad \quad \text{value} (\text{cadr} (\text{call-actuals} (stmt)), \\
& \quad \quad \text{bindings} (\text{top} (ctrl-stk)))), \\
& \quad \text{cons} ('value, \\
& \quad \quad \text{value} (\text{caddr} (\text{call-actuals} (stmt)), \\
& \quad \quad \text{bindings} (\text{top} (ctrl-stk)))), \\
& \quad \text{cons} ('array-size, \\
& \quad \quad \text{tag} ('int, \\
& \quad \quad \text{caddr} (\text{call-actuals} (stmt))), \\
& \quad \text{cons} ('temp-i, \\
& \quad \quad \text{mg-to-p-simple-literal} (\text{caddr} (\text{assoc} (\text{cadr} (\text{call-actuals} (stmt)), \\
& \quad \quad \quad \text{mg-alist} (mg-state)))))), \\
& \quad \text{tag} ('pc, \\
& \quad \quad \text{cons} (subr, \text{length} (\text{code} (cinfo)) \\
& \quad \quad \quad + 5))), \\
& \quad ctrl-stk), \\
& \text{push} (\text{value} ('value, \\
& \quad \text{list} (\text{cons} ('a, \\
& \quad \quad \text{value} (\text{car} (\text{call-actuals} (stmt)), \\
& \quad \quad \text{bindings} (\text{top} (ctrl-stk))), \\
& \quad \text{cons} ('i, \\
& \quad \quad \text{value} (\text{cadr} (\text{call-actuals} (stmt)), \\
& \quad \quad \text{bindings} (\text{top} (ctrl-stk))), \\
& \quad \text{cons} ('value, \\
& \quad \quad \text{value} (\text{caddr} (\text{call-actuals} (stmt)), \\
& \quad \quad \text{bindings} (\text{top} (ctrl-stk))), \\
& \quad \text{cons} ('array-size, \\
& \quad \quad \text{tag} ('int, \\
& \quad \quad \text{caddr} (\text{call-actuals} (stmt))), \\
& \quad \text{cons} ('temp-i, \\
& \quad \quad \text{mg-to-p-simple-literal} (\text{caddr} (\text{assoc} (\text{cadr} (\text{call-actuals} (stmt)), \\
& \quad \quad \quad \text{mg-alist} (mg-state)))))), \\
& \quad \text{map-down-values} (\text{mg-alist} (mg-state), \\
& \quad \quad \text{bindings} (\text{top} (ctrl-stk)), \\
& \quad \quad temp-stk)), \\
& \text{translate-proc-list} (proc-list),
\end{aligned}$$

```

list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
= p-state (tag ('pc,
               '(mg-array-element-assignment . 10)),
          push (p-frame (list (cons ('a,
                                     value (car (call-actuals (stmt)),
                                     bindings (top (ctrl-stk)))),
                                cons ('i,
                                     value (cadr (call-actuals (stmt)),
                                     bindings (top (ctrl-stk)))),
                                cons ('value,
                                     value (caddr (call-actuals (stmt)),
                                     bindings (top (ctrl-stk)))),
                                cons ('array-size,
                                     tag ('int,
                                     caddr (call-actuals (stmt)))),
                                cons ('temp-i,
                                     mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                     mg-alist (mg-state))))))),
               tag ('pc,
                   cons (subr, length (code (cinfo))
                       + 5))),
          ctrl-stk),
push (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                           mg-alist (mg-state)))),
    map-down-values (mg-alist (mg-state),
                     bindings (top (ctrl-stk)),
                     temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-array-element-assignment-steps-16-17-no-error

$((n \neq 0)$
 $\wedge \quad (\neg \text{resources-inadequatep} (stmt,$
 $\quad \quad \quad proc-list,$

```

                                list (length (temp-stk),
                                      p-ctrl-stk-size (ctrl-stk))))
^ (car (stmt) = 'predefined-proc-call-mg)
^ (call-name (stmt) = 'mg-array-element-assignment)
^ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
^ ok-mg-def-plistp (proc-list)
^ ok-mg-statep (mg-state, r-cond-list)
^ (code (translate-def-body (assoc (subr, proc-list), proc-list))
      = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                code2))
^ user-defined-procp (subr, proc-list)
^ listp (ctrl-stk)
^ all-cars-unique (mg-alist (mg-state))
^ signatures-match (mg-alist (mg-state), name-alist)
^ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                              bindings (top (ctrl-stk)),
                              temp-stk)
^ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
^ normal (mg-state)
^ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                            mg-alist (mg-state)))))))
^ (idifference (caddr (call-actuals (stmt)),
               untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                            mg-alist (mg-state))))))) ≠ 0))
→ (p-step (p-step (p-state (tag ('pc,
                                '(mg-array-element-assignment
                                  . 10)),
                                push (p-frame (list (cons ('a,
                                                            value (car (call-actuals (stmt)),
                                                            bindings (top (ctrl-stk)))),
                                                            cons ('i,
                                                            value (cadr (call-actuals (stmt)),
                                                            bindings (top (ctrl-stk)))),
                                                            cons ('value,
                                                            value (caddr (call-actuals (stmt)),
                                                            bindings (top (ctrl-stk)))),
                                                            cons ('array-size,
                                                            tag ('int,
                                                            caddr (call-actuals (stmt)))),
                                                            cons ('temp-i,
                                                            mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (s
                                                            mg-alist (mg-state)),
                                                            tag ('pc,
                                                            cons (subr,

```

```

length (code (cinfo))
+ 5))),
ctrl-stk),
push (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
mg-alist (mg-state))))),
map-down-values (mg-alist (mg-state),
bindings (top (ctrl-stk)),
temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
mg-cond-to-p-nat (cc (mg-state),
t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run)))
= p-state (tag ('pc,
'(mg-array-element-assignment . 12)),
push (p-frame (list (cons ('a,
value (car (call-actuals (stmt)),
bindings (top (ctrl-stk))))),
cons ('i,
value (cadr (call-actuals (stmt)),
bindings (top (ctrl-stk))))),
cons ('value,
value (caddr (call-actuals (stmt)),
bindings (top (ctrl-stk))))),
cons ('array-size,
tag ('int,
caddr (call-actuals (stmt))))),
cons ('temp-i,
mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
mg-alist (mg-state))))))),
tag ('pc,
cons (subr, length (code (cinfo))
+ 5))),
ctrl-stk),
push (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
mg-alist (mg-state))))),
push (value (car (call-actuals (stmt)),
bindings (top (ctrl-stk))),
push (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
mg-alist (mg-state))))),
map-down-values (mg-alist (mg-state),

```

bindings (top (ctrl-stk)),
temp-stk))))),

translate-proc-list (proc-list),
list (list ('c-c,
 mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: mg-array-element-assignment-step-18-no-error

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
 proc-list,
 list (length (temp-stk),
 p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-array-element-assignment)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp (proc-list)
 ∧ ok-mg-statep (mg-state, r-cond-list)
 ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
 = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
 code2))
 ∧ user-defined-procp (subr, proc-list)
 ∧ listp (ctrl-stk)
 ∧ all-cars-unique (mg-alist (mg-state))
 ∧ signatures-match (mg-alist (mg-state), name-alist)
 ∧ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
 bindings (top (ctrl-stk)),
 temp-stk)
 ∧ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 ∧ normal (mg-state)
 ∧ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
 mg-alist (mg-state)))))))
 ∧ (idifference (caddr (call-actuals (stmt)),
 untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
 mg-alist (mg-state))))))) ≠ 0))
 → (p-step (p-state (tag ('pc,
 'mg-array-element-assignment . 12)),
 push (p-frame (list (cons ('a,
 value (car (call-actuals (stmt)),
 bindings (top (ctrl-stk)))),
 cons ('i,

```

value (cadr (call-actuals (stmt)),
bindings (top (ctrl-stk))),
cons ('value,
value (caddr (call-actuals (stmt)),
bindings (top (ctrl-stk))),
cons ('array-size,
tag ('int,
caddr (call-actuals (stmt))),
cons ('temp-i,
mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
mg-alist (mg-state)))))),
tag ('pc,
cons (subr, length (code (cinfo))
+ 5))),
ctrl-stk),
push (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
mg-alist (mg-state))))),
push (value (car (call-actuals (stmt)),
bindings (top (ctrl-stk))),
push (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
mg-alist (mg-state))))),
map-down-values (mg-alist (mg-state),
bindings (top (ctrl-stk)),
temp-stk))),
translate-proc-list (proc-list),
list (list ('c-c,
mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
= p-state (tag ('pc,
'(mg-array-element-assignment . 13)),
push (p-frame (list (cons ('a,
value (car (call-actuals (stmt)),
bindings (top (ctrl-stk))),
cons ('i,
value (cadr (call-actuals (stmt)),
bindings (top (ctrl-stk))),
cons ('value,
value (caddr (call-actuals (stmt)),
bindings (top (ctrl-stk))),
cons ('array-size,
tag ('int,

```

```

                                caddr (call-actuals (stmt))),
                                cons ('temp-i,
                                      mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                           mg-alist (mg-state)))))),
                                tag ('pc,
                                      cons (subr, length (code (cinfo)
                                                                + 5))),
                                ctrl-stk),
                                push (tag ('nat,
                                           untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                    mg-alist (mg-state)))))),
                                      push (value (car (call-actuals (stmt)),
                                                    bindings (top (ctrl-stk))),
                                      push (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                    mg-alist (mg-state))))),
                                      map-down-values (mg-alist (mg-state),
                                                            bindings (top (ctrl-stk),
                                                                temp-stk))),
                                translate-proc-list (proc-list),
                                list (list ('c-c,
                                           mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                                MG-MAX-CTRL-STK-SIZE,
                                MG-MAX-TEMP-STK-SIZE,
                                MG-WORD-SIZE,
                                'run))

```

THEOREM: mg-array-element-assignment-step-19-no-error

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
                              proc-list,
                              list (length (temp-stk),
                                      p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-array-element-assignment)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-mg-def-plistp (proc-list)
 ∧ ok-mg-statep (mg-state, r-cond-list)
 ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
              code2))
 ∧ user-defined-procp (subr, proc-list)
 ∧ listp (ctrl-stk)
 ∧ all-cars-unique (mg-alist (mg-state))
 ∧ signatures-match (mg-alist (mg-state), name-alist)

```

```

 $\wedge$   mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                     bindings (top (ctrl-stk)),
                                     temp-stk)
 $\wedge$   no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 $\wedge$   normal (mg-state)
 $\wedge$   ( $\neg$  negativep (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                mg-alist (mg-state)))))))
 $\wedge$   (idifference (caddr (call-actuals (stmt)),
                    untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                mg-alist (mg-state))))))  $\neq$  0))
 $\rightarrow$   (p-step (p-state (tag ('pc,
                            '(mg-array-element-assignment . 13)),
                            push (p-frame (list (cons ('a,
                                                        value (car (call-actuals (stmt)),
                                                        bindings (top (ctrl-stk))),
                                                        cons ('i,
                                                            value (cadr (call-actuals (stmt)),
                                                            bindings (top (ctrl-stk))),
                                                            cons ('value,
                                                                value (caddr (call-actuals (stmt)),
                                                                bindings (top (ctrl-stk))),
                                                                cons ('array-size,
                                                                    tag ('int,
                                                                        caddr (call-actuals (stmt))),
                                                                    cons ('temp-i,
                                                                        mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                                                    mg-alist (mg-state)))))),
                                                                    tag ('pc,
                                                                        cons (subr, length (code (cinfo))
                                                                                                    + 5))),
                                                                    ctrl-stk),
                                                                push (tag ('nat,
                                                                    untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                                                    mg-alist (mg-state)))))),
                                                                    push (value (car (call-actuals (stmt)),
                                                                    bindings (top (ctrl-stk))),
                                                                    push (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                                                    mg-alist (mg-state))))),
                                                                    map-down-values (mg-alist (mg-state),
                                                                    bindings (top (ctrl-stk)),
                                                                    temp-stk))),
                                                                translate-proc-list (proc-list),
                                                                list (list ('c-c,
                                                                    mg-cond-to-p-nat (cc (mg-state), t-cond-list))),

```

```

MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
= p-state (tag ('pc,
               '(mg-array-element-assignment . 14)),
  push (p-frame (list (cons ('a,
                             value (car (call-actuals (stmt)),
                             bindings (top (ctrl-stk))))),
               cons ('i,
                     value (cadr (call-actuals (stmt)),
                     bindings (top (ctrl-stk))))),
               cons ('value,
                     value (caddr (call-actuals (stmt)),
                     bindings (top (ctrl-stk))))),
               cons ('array-size,
                     tag ('int,
                           caddr (call-actuals (stmt))))),
               cons ('temp-i,
                     mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                             mg-alist (mg-state)))))),
               tag ('pc,
                   cons (subr, length (code (cinfo))
                       + 5))),
  ctrl-stk),
  push (tag ('nat,
            untag (value (car (call-actuals (stmt)),
                          bindings (top (ctrl-stk))))
      + untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                    mg-alist (mg-state)))))),
    push (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                mg-alist (mg-state))))),
      map-down-values (mg-alist (mg-state),
                        bindings (top (ctrl-stk)),
                        temp-stk))),
  translate-proc-list (proc-list),
  list (list ('c-c,
             mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
  MG-MAX-CTRL-STK-SIZE,
  MG-MAX-TEMP-STK-SIZE,
  MG-WORD-SIZE,
  'run))

```

THEOREM: mg-array-element-assignment-index-lessp-temp-stk-length

$$\begin{aligned}
& ((\text{car}(stmt) = \text{'predefined-proc-call-mg'}) \\
& \wedge (\text{call-name}(stmt) = \text{'mg-array-element-assignment'}) \\
& \wedge \text{ok-mg-statement}(stmt, r\text{-cond-list}, name\text{-alist}, proc\text{-list}) \\
& \wedge \text{ok-mg-statep}(mg\text{-state}, r\text{-cond-list}) \\
& \wedge \text{mg-vars-list-ok-in-p-state}(mg\text{-alist}(mg\text{-state}), \\
& \quad \text{bindings}(\text{top}(ctrl\text{-stk})), \\
& \quad \text{temp-stk}) \\
& \wedge \text{signatures-match}(mg\text{-alist}(mg\text{-state}), name\text{-alist}) \\
& \wedge (\neg \text{negativep}(\text{untag}(\text{caddr}(\text{assoc}(\text{cadr}(\text{call-actuals}(stmt)), \\
& \quad \text{mg-alist}(mg\text{-state})))))) \\
& \wedge (\text{idifference}(\text{caddr}(\text{call-actuals}(stmt)), \\
& \quad \text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{cadr}(\text{call-actuals}(stmt)), \\
& \quad \text{mg-alist}(mg\text{-state})))))) \neq 0)) \\
\rightarrow & (((\text{untag}(\text{value}(\text{car}(\text{call-actuals}(stmt))), \text{bindings}(\text{top}(ctrl\text{-stk})))) \\
& \quad + \text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{cadr}(\text{call-actuals}(stmt)), \\
& \quad \text{mg-alist}(mg\text{-state})))))) \\
& \quad < \text{length}(\text{temp-stk})) \\
& = \mathbf{t})
\end{aligned}$$

THEOREM: mg-array-element-assignment-step-20-no-error

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge (\neg \text{resources-inadequatep}(stmt, \\
& \quad \text{proc-list}, \\
& \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \text{p-ctrl-stk-size}(ctrl\text{-stk})))) \\
& \wedge (\text{car}(stmt) = \text{'predefined-proc-call-mg'}) \\
& \wedge (\text{call-name}(stmt) = \text{'mg-array-element-assignment'}) \\
& \wedge \text{ok-mg-statement}(stmt, r\text{-cond-list}, name\text{-alist}, proc\text{-list}) \\
& \wedge \text{ok-mg-def-plistp}(proc\text{-list}) \\
& \wedge \text{ok-mg-statep}(mg\text{-state}, r\text{-cond-list}) \\
& \wedge (\text{code}(\text{translate-def-body}(\text{assoc}(subr, proc\text{-list}), proc\text{-list})) \\
& \quad = \text{append}(\text{code}(\text{translate}(cinfo, t\text{-cond-list}, stmt, proc\text{-list})), \\
& \quad \text{code2})) \\
& \wedge \text{user-defined-procp}(subr, proc\text{-list}) \\
& \wedge \text{listp}(ctrl\text{-stk}) \\
& \wedge \text{all-cars-unique}(mg\text{-alist}(mg\text{-state})) \\
& \wedge \text{signatures-match}(mg\text{-alist}(mg\text{-state}), name\text{-alist}) \\
& \wedge \text{mg-vars-list-ok-in-p-state}(mg\text{-alist}(mg\text{-state}), \\
& \quad \text{bindings}(\text{top}(ctrl\text{-stk})), \\
& \quad \text{temp-stk}) \\
& \wedge \text{no-p-aliasing}(\text{bindings}(\text{top}(ctrl\text{-stk})), mg\text{-alist}(mg\text{-state})) \\
& \wedge \text{normal}(mg\text{-state}) \\
& \wedge (\neg \text{negativep}(\text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{cadr}(\text{call-actuals}(stmt)), \\
& \quad \text{mg-alist}(mg\text{-state}))))))
\end{aligned}$$

$$\begin{aligned}
& \wedge \quad (\text{idifference} (\text{caddr} (\text{call-actuals} (stmt)), \\
& \quad \text{untag} (\text{mg-to-p-simple-literal} (\text{caddr} (\text{assoc} (\text{cadr} (\text{call-actuals} (stmt)), \\
& \quad \quad \text{mg-alist} (mg-state)))))) \neq 0)) \\
\rightarrow & \quad (\text{p-step} (\text{p-state} (\text{tag} ('pc, \\
& \quad \quad \text{'(mg-array-element-assignment . 14)}, \\
& \quad \quad \text{push} (\text{p-frame} (\text{list} (\text{cons} ('a, \\
& \quad \quad \quad \text{value} (\text{car} (\text{call-actuals} (stmt)), \\
& \quad \quad \quad \text{bindings} (\text{top} (ctrl-stk)))), \\
& \quad \quad \quad \text{cons} ('i, \\
& \quad \quad \quad \text{value} (\text{cadr} (\text{call-actuals} (stmt)), \\
& \quad \quad \quad \text{bindings} (\text{top} (ctrl-stk)))), \\
& \quad \quad \quad \text{cons} ('value, \\
& \quad \quad \quad \text{value} (\text{caddr} (\text{call-actuals} (stmt)), \\
& \quad \quad \quad \text{bindings} (\text{top} (ctrl-stk)))), \\
& \quad \quad \quad \text{cons} ('array-size, \\
& \quad \quad \quad \text{tag} ('int, \\
& \quad \quad \quad \text{caddr} (\text{call-actuals} (stmt))), \\
& \quad \quad \quad \text{cons} ('temp-i, \\
& \quad \quad \quad \text{mg-to-p-simple-literal} (\text{caddr} (\text{assoc} (\text{cadr} (\text{call-actuals} (stmt)), \\
& \quad \quad \quad \text{mg-alist} (mg-state)))))), \\
& \quad \quad \text{tag} ('pc, \\
& \quad \quad \text{cons} (subr, \text{length} (\text{code} (cinfo)) \\
& \quad \quad \quad + 5))), \\
& \quad \quad ctrl-stk), \\
& \quad \text{push} (\text{tag} ('nat, \\
& \quad \quad \text{untag} (\text{value} (\text{car} (\text{call-actuals} (stmt)), \\
& \quad \quad \quad \text{bindings} (\text{top} (ctrl-stk)))) \\
& \quad \quad + \text{untag} (\text{mg-to-p-simple-literal} (\text{caddr} (\text{assoc} (\text{cadr} (\text{call-actuals} (stmt)), \\
& \quad \quad \quad \text{mg-alist} (mg-state)))))), \\
& \quad \quad \text{push} (\text{mg-to-p-simple-literal} (\text{caddr} (\text{assoc} (\text{caddr} (\text{call-actuals} (stmt)), \\
& \quad \quad \quad \text{mg-alist} (mg-state))))), \\
& \quad \quad \text{map-down-values} (\text{mg-alist} (mg-state), \\
& \quad \quad \quad \text{bindings} (\text{top} (ctrl-stk)), \\
& \quad \quad \quad temp-stk))), \\
& \quad \text{translate-proc-list} (proc-list), \\
& \quad \text{list} (\text{list} ('c-c, \\
& \quad \quad \text{mg-cond-to-p-nat} (\text{cc} (mg-state), t-cond-list))), \\
& \quad \text{MG-MAX-CTRL-STK-SIZE}, \\
& \quad \text{MG-MAX-TEMP-STK-SIZE}, \\
& \quad \text{MG-WORD-SIZE}, \\
& \quad 'run)) \\
= & \quad \text{p-state} (\text{tag} ('pc, \\
& \quad \quad \text{'(mg-array-element-assignment . 15)}, \\
& \quad \quad \text{push} (\text{p-frame} (\text{list} (\text{cons} ('a,
\end{aligned}$$

```

value (car (call-actuals (stmt)),
      bindings (top (ctrl-stk)))),
cons ('i,
      value (cadr (call-actuals (stmt)),
            bindings (top (ctrl-stk)))),
cons ('value,
      value (caddr (call-actuals (stmt)),
            bindings (top (ctrl-stk)))),
cons ('array-size,
      tag ('int,
            caddr (call-actuals (stmt)))),
cons ('temp-i,
      mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                           mg-alist (mg-state)))))),
tag ('pc,
     cons (subr, length (code (cinfo)
                               + 5))),
ctrl-stk),
rput (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                           mg-alist (mg-state)))),
      untag (value (car (call-actuals (stmt)),
                    bindings (top (ctrl-stk))))
+ untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                           mg-alist (mg-state)))))),
      map-down-values (mg-alist (mg-state),
                             bindings (top (ctrl-stk)),
                             temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-array-element-assignment-steps-21-22-no-error

```

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
                             proc-list,
                             list (length (temp-stk),
                                       p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'predefined-proc-call-mg)
 ∧ (call-name (stmt) = 'mg-array-element-assignment)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)

```

```

^ ok-mg-def-plistp (proc-list)
^ ok-mg-statep (mg-state, r-cond-list)
^ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
              code2))
^ user-defined-procp (subr, proc-list)
^ listp (ctrl-stk)
^ all-cars-unique (mg-alist (mg-state))
^ signatures-match (mg-alist (mg-state), name-alist)
^ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                     bindings (top (ctrl-stk)),
                                     temp-stk)
^ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
^ normal (mg-state)
^ (¬ negativep (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                             mg-alist (mg-state)))))))
^ (idifference (caddr (call-actuals (stmt)),
               untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                             mg-alist (mg-state)))))) ≠ 0))
→ (p-step (p-step (p-state (tag ('pc,
                                '(mg-array-element-assignment
                                  . 15)),
                                push (p-frame (list (cons ('a,
                                                            value (car (call-actuals (stmt)),
                                                            bindings (top (ctrl-stk))),
                                                            cons ('i,
                                                            value (cadr (call-actuals (stmt)),
                                                            bindings (top (ctrl-stk))),
                                                            cons ('value,
                                                            value (caddr (call-actuals (stmt)),
                                                            bindings (top (ctrl-stk))),
                                                            cons ('array-size,
                                                            tag ('int,
                                                            caddr (call-actuals (stmt))),
                                                            cons ('temp-i,
                                                            mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                            mg-alist (mg-state)),
                                                            tag ('pc,
                                                            cons (subr,
                                                            length (code (cinfo))
                                                            + 5))),
                                                            ctrl-stk),
                                rput (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                            mg-alist (mg-state))))),

```

$$\begin{aligned}
& \text{untag}(\text{value}(\text{car}(\text{call-actuals}(stmt)), \\
& \quad \text{bindings}(\text{top}(ctrl-stk)))) \\
& + \text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{cadr}(\text{call-actuals}(stmt)), \\
& \quad \text{mg-alist}(mg-state))))), \\
& \text{map-down-values}(\text{mg-alist}(mg-state), \\
& \quad \text{bindings}(\text{top}(ctrl-stk)), \\
& \quad \text{temp-stk}), \\
& \text{translate-proc-list}(proc-list), \\
& \text{list}(\text{list}('c-c, \\
& \quad \text{mg-cond-to-p-nat}(\text{cc}(mg-state), \\
& \quad \quad t-cond-list))), \\
& \text{MG-MAX-CTRL-STK-SIZE}, \\
& \text{MG-MAX-TEMP-STK-SIZE}, \\
& \text{MG-WORD-SIZE}, \\
& 'run))) \\
= & \text{p-state}(\text{tag}('pc, \text{cons}(subr, \text{length}(\text{code}(cinfo)) + 5)), \\
& \quad ctrl-stk, \\
& \text{rput}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{caddr}(\text{call-actuals}(stmt)), \\
& \quad \text{mg-alist}(mg-state))))), \\
& \text{untag}(\text{value}(\text{car}(\text{call-actuals}(stmt)), \\
& \quad \text{bindings}(\text{top}(ctrl-stk)))) \\
& + \text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{cadr}(\text{call-actuals}(stmt)), \\
& \quad \text{mg-alist}(mg-state))))), \\
& \text{map-down-values}(\text{mg-alist}(mg-state), \\
& \quad \text{bindings}(\text{top}(ctrl-stk)), \\
& \quad \text{temp-stk}), \\
& \text{translate-proc-list}(proc-list), \\
& \text{list}(\text{list}('c-c, \\
& \quad \text{mg-cond-to-p-nat}(\text{cc}(mg-state), t-cond-list))), \\
& \text{MG-MAX-CTRL-STK-SIZE}, \\
& \text{MG-MAX-TEMP-STK-SIZE}, \\
& \text{MG-WORD-SIZE}, \\
& 'run))
\end{aligned}$$

THEOREM: mg-array-element-assignment-push-cc

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge (\neg \text{resources-inadequatep}(stmt, \\
& \quad \quad \quad proc-list, \\
& \quad \quad \quad \text{list}(\text{length}(temp-stk), \\
& \quad \quad \quad \text{p-ctrl-stk-size}(ctrl-stk)))) \\
& \wedge (\text{car}(stmt) = 'predefined-proc-call-mg) \\
& \wedge (\text{call-name}(stmt) = 'mg-array-element-assignment) \\
& \wedge \text{ok-mg-statement}(stmt, r-cond-list, name-alist, proc-list) \\
& \wedge \text{ok-mg-def-plistp}(proc-list)
\end{aligned}$$

$$\begin{aligned}
& \wedge \text{ok-mg-statep}(mg\text{-state}, r\text{-cond-list}) \\
& \wedge (\text{code}(\text{translate-def-body}(\text{assoc}(subr, proc\text{-list}), proc\text{-list})) \\
& \quad = \text{append}(\text{code}(\text{translate}(cinfo, t\text{-cond-list}, stmt, proc\text{-list})), \\
& \quad \quad \text{code2})) \\
& \wedge \text{user-defined-procp}(subr, proc\text{-list}) \\
& \wedge \text{listp}(ctrl\text{-stk}) \\
& \wedge \text{all-cars-unique}(mg\text{-alist}(mg\text{-state})) \\
& \wedge \text{signatures-match}(mg\text{-alist}(mg\text{-state}), name\text{-alist}) \\
& \wedge \text{normal}(mg\text{-state}) \\
\rightarrow & (\text{p-step}(\text{p-state}(\text{tag}('pc, \text{cons}(subr, \text{length}(\text{code}(cinfo)) + 5)), \\
& \quad ctrl\text{-stk}, \\
& \quad temp\text{-stk}, \\
& \quad \text{translate-proc-list}(proc\text{-list}), \\
& \quad \text{list}(\text{list}('c-c, cc\text{-value})), \\
& \quad \text{MG-MAX-CTRL-STK-SIZE}, \\
& \quad \text{MG-MAX-TEMP-STK-SIZE}, \\
& \quad \text{MG-WORD-SIZE}, \\
& \quad 'run)) \\
& = \text{p-state}(\text{tag}('pc, \text{cons}(subr, \text{length}(\text{code}(cinfo)) + 6)), \\
& \quad ctrl\text{-stk}, \\
& \quad \text{push}(cc\text{-value}, temp\text{-stk}), \\
& \quad \text{translate-proc-list}(proc\text{-list}), \\
& \quad \text{list}(\text{list}('c-c, cc\text{-value})), \\
& \quad \text{MG-MAX-CTRL-STK-SIZE}, \\
& \quad \text{MG-MAX-TEMP-STK-SIZE}, \\
& \quad \text{MG-WORD-SIZE}, \\
& \quad 'run))
\end{aligned}$$

THEOREM: mg-array-element-assignment-sub1-cc

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge (\neg \text{resources-inadequatep}(stmt, \\
& \quad \quad \quad proc\text{-list}, \\
& \quad \quad \quad \text{list}(\text{length}(temp\text{-stk}), \\
& \quad \quad \quad \text{p-ctrl-stk-size}(ctrl\text{-stk})))) \\
& \wedge (\text{car}(stmt) = 'predefined-proc-call-mg) \\
& \wedge (\text{call-name}(stmt) = 'mg-array-element-assignment) \\
& \wedge \text{ok-mg-statement}(stmt, r\text{-cond-list}, name\text{-alist}, proc\text{-list}) \\
& \wedge \text{ok-mg-def-plistp}(proc\text{-list}) \\
& \wedge \text{ok-mg-statep}(mg\text{-state}, r\text{-cond-list}) \\
& \wedge (\text{code}(\text{translate-def-body}(\text{assoc}(subr, proc\text{-list}), proc\text{-list})) \\
& \quad = \text{append}(\text{code}(\text{translate}(cinfo, t\text{-cond-list}, stmt, proc\text{-list})), \\
& \quad \quad \text{code2})) \\
& \wedge \text{user-defined-procp}(subr, proc\text{-list}) \\
& \wedge \text{listp}(ctrl\text{-stk})
\end{aligned}$$

$$\begin{aligned}
& \wedge \text{all-cars-unique}(\text{mg-alist}(\text{mg-state})) \\
& \wedge \text{signatures-match}(\text{mg-alist}(\text{mg-state}), \text{name-alist}) \\
& \wedge \text{normal}(\text{mg-state}) \\
& \wedge (\text{cc-value} \in \text{list}('(\text{nat } 1), '(\text{nat } 2)))) \\
\rightarrow & (\text{p-step}(\text{p-state}(\text{tag}('pc, \text{cons}(\text{subr}, \text{length}(\text{code}(\text{cinfo})) + 6)), \\
& \quad \text{ctrl-stk}, \\
& \quad \text{push}(\text{cc-value}, \text{temp-stk}), \\
& \quad \text{translate-proc-list}(\text{proc-list}), \\
& \quad \text{list}(\text{list}('c-c, \text{cc-value})), \\
& \quad \text{MG-MAX-CTRL-STK-SIZE}, \\
& \quad \text{MG-MAX-TEMP-STK-SIZE}, \\
& \quad \text{MG-WORD-SIZE}, \\
& \quad 'run)) \\
= & \text{p-state}(\text{tag}('pc, \text{cons}(\text{subr}, \text{length}(\text{code}(\text{cinfo})) + 7)), \\
& \quad \text{ctrl-stk}, \\
& \quad \text{push}(\text{tag}('nat, \text{untag}(\text{cc-value}) - 1), \text{temp-stk}), \\
& \quad \text{translate-proc-list}(\text{proc-list}), \\
& \quad \text{list}(\text{list}('c-c, \text{cc-value})), \\
& \quad \text{MG-MAX-CTRL-STK-SIZE}, \\
& \quad \text{MG-MAX-TEMP-STK-SIZE}, \\
& \quad \text{MG-WORD-SIZE}, \\
& \quad 'run))
\end{aligned}$$

THEOREM: mg-array-element-assignment-last-step-error-case

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge (\neg \text{resources-inadequatep}(\text{stmt}, \\
& \quad \text{proc-list}, \\
& \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \quad \text{p-ctrl-stk-size}(\text{ctrl-stk})))) \\
& \wedge (\text{car}(\text{stmt}) = \text{'predefined-proc-call-mg}) \\
& \wedge (\text{call-name}(\text{stmt}) = \text{'mg-array-element-assignment}) \\
& \wedge \text{ok-mg-statement}(\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}) \\
& \wedge \text{ok-mg-def-plistp}(\text{proc-list}) \\
& \wedge \text{ok-mg-statep}(\text{mg-state}, \text{r-cond-list}) \\
& \wedge (\text{code}(\text{translate-def-body}(\text{assoc}(\text{subr}, \text{proc-list}), \text{proc-list})) \\
& \quad = \text{append}(\text{code}(\text{translate}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list})), \\
& \quad \quad \text{code2})) \\
& \wedge \text{user-defined-procp}(\text{subr}, \text{proc-list}) \\
& \wedge \text{listp}(\text{ctrl-stk}) \\
& \wedge \text{all-cars-unique}(\text{mg-alist}(\text{mg-state})) \\
& \wedge \text{signatures-match}(\text{mg-alist}(\text{mg-state}), \text{name-alist}) \\
& \wedge \text{mg-vars-list-ok-in-p-state}(\text{mg-alist}(\text{mg-state}), \\
& \quad \text{bindings}(\text{top}(\text{ctrl-stk}), \\
& \quad \text{temp-stk}))
\end{aligned}$$

$$\begin{aligned}
& \wedge \text{no-p-aliasing}(\text{bindings}(\text{top}(\text{ctrl-stk})), \text{mg-alist}(\text{mg-state})) \\
& \wedge \text{normal}(\text{mg-state}) \\
& \wedge (\text{negativep}(\text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{cadr}(\text{call-actuals}(\text{stmt})), \\
& \hspace{15em} \text{mg-alist}(\text{mg-state})))))) \\
& \vee (\text{idifference}(\text{caddr}(\text{call-actuals}(\text{stmt})), \\
& \hspace{10em} \text{untag}(\text{mg-to-p-simple-literal}(\text{caddr}(\text{assoc}(\text{cadr}(\text{call-actuals}(\text{stmt})), \\
& \hspace{15em} \text{mg-alist}(\text{mg-state})))))) \simeq 0))) \\
\rightarrow & (\text{p-step}(\text{p-state}(\text{tag}(\text{'pc}, \text{cons}(\text{subr}, \text{length}(\text{code}(\text{cinfo})) + 7)), \\
& \hspace{10em} \text{ctrl-stk}, \\
& \text{push}(\text{tag}(\text{'nat}, \text{untag}(\text{'(nat 1)}) - 1), \\
& \hspace{10em} \text{map-down-values}(\text{mg-alist}(\text{mg-state}), \\
& \hspace{10em} \text{bindings}(\text{top}(\text{ctrl-stk})), \\
& \hspace{10em} \text{temp-stk})), \\
& \text{translate-proc-list}(\text{proc-list}), \\
& \text{'((c-c (nat 1))}), \\
& \text{MG-MAX-CTRL-STK-SIZE}, \\
& \text{MG-MAX-TEMP-STK-SIZE}, \\
& \text{MG-WORD-SIZE}, \\
& \text{'run})) \\
= & \text{p-state}(\text{tag}(\text{'pc}, \\
& \hspace{10em} \text{cons}(\text{subr}, \\
& \hspace{10em} \text{if normal}(\text{mg-meaning-r}(\text{stmt}, \\
& \hspace{10em} \text{proc-list}, \\
& \hspace{10em} \text{mg-state}, \\
& \hspace{10em} n, \\
& \hspace{10em} \text{list}(\text{length}(\text{temp-stk}), \\
& \hspace{10em} \text{p-ctrl-stk-size}(\text{ctrl-stk})))) \\
& \text{then length}(\text{code}(\text{translate}(\text{cinfo}, \\
& \hspace{10em} \text{t-cond-list}, \\
& \hspace{10em} \text{stmt}, \\
& \hspace{10em} \text{proc-list}))) \\
& \text{else find-label}(\text{fetch-label}(\text{cc}(\text{mg-meaning-r}(\text{stmt}, \\
& \hspace{10em} \text{proc-list}, \\
& \hspace{10em} \text{mg-state}, \\
& \hspace{10em} n, \\
& \hspace{10em} \text{list}(\text{length}(\text{temp-stk}), \\
& \hspace{10em} \text{p-ctrl-stk-size}(\text{ctrl-stk}))))), \\
& \hspace{10em} \text{label-alist}(\text{translate}(\text{cinfo}, \\
& \hspace{10em} \text{t-cond-list}, \\
& \hspace{10em} \text{stmt}, \\
& \hspace{10em} \text{proc-list}))), \\
& \hspace{10em} \text{append}(\text{code}(\text{translate}(\text{cinfo}, \\
& \hspace{10em} \text{t-cond-list}, \\
& \hspace{10em} \text{stmt},
\end{aligned}$$

```

                                proc-list)),
                                code2)) endif)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (stmt,
                                proc-list,
                                mg-state,
                                n,
                                list (length (temp-stk),
                                        p-ctrl-stk-size (ctrl-stk)))))
                                bindings (top (ctrl-stk)),
                                temp-stk),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-meaning-r (stmt,
                                proc-list,
                                mg-state,
                                n,
                                list (length (temp-stk),
                                        p-ctrl-stk-size (ctrl-stk)))))
                                t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: put-preserves-ok-mg-array-value

```

(ok-mg-array-value (z, type)
  ∧ simple-typed-literalp (x, array-elemtype (type))
  ∧ (i < length (z)))
→ ok-mg-array-value (put (x, i, z), type)

```

THEOREM: arrays-have-ok-values

```

(mg-alistp (mg-alist) ∧ array-identifierp (a, mg-alist))
→ ok-mg-array-value (caddr (assoc (a, mg-alist)), cadr (assoc (a, mg-alist)))

```

DEFINITION:

put-deposit-array-value-induction-hint (lst, nat, temp-stk, index)

```

= if index ≃ 0 then t
  else put-deposit-array-value-induction-hint (cdr (lst),
                                                add1-nat (nat),
                                                deposit-temp (mg-to-p-simple-literal (car (lst)),
                                                            nat,
                                                            temp-stk),
                                                index - 1) endif

```

; I think the splits over whether (nlistp (cdr lst)) are not necessary

THEOREM: put-deposit-array-value-rewrite

$$\begin{aligned}
& ((index < \text{length}(lst)) \\
& \wedge ((\text{untag}(nat) + (\text{length}(lst) - 1)) < \text{length}(temp\text{-}stk)) \\
& \wedge (\text{untag}(nat) \in \mathbf{N})) \\
& \rightarrow (\text{deposit-array-value}(\text{put}(value, index, lst), nat, temp\text{-}stk) \\
& \quad = \text{rput}(\text{mg-to-p-simple-literal}(value), \\
& \quad \quad \text{untag}(nat) + index, \\
& \quad \quad \text{deposit-array-value}(lst, nat, temp\text{-}stk)))
\end{aligned}$$

EVENT: Disable put-deposit-array-value-rewrite.

THEOREM: mg-array-element-assignment-deposit-array-value-rewrite

$$\begin{aligned}
& (\text{all-cars-unique}(mg\text{-}vars) \\
& \wedge \text{mg-alistp}(mg\text{-}vars) \\
& \wedge \text{mg-vars-list-ok-in-p-state}(mg\text{-}vars, bindings, temp\text{-}stk) \\
& \wedge \text{no-p-aliasing}(bindings, mg\text{-}vars) \\
& \wedge (index < \text{array-length}(\text{cadr}(\text{assoc}(a, mg\text{-}vars)))) \\
& \wedge \text{array-identifierp}(a, mg\text{-}vars)) \\
& \rightarrow (\text{deposit-array-value}(\text{put}(value, index, \text{caddr}(\text{assoc}(a, mg\text{-}vars))), \\
& \quad \quad \text{cdr}(\text{assoc}(a, bindings)), \\
& \quad \quad \text{map-down-values}(mg\text{-}vars, bindings, temp\text{-}stk)) \\
& \quad = \text{rput}(\text{mg-to-p-simple-literal}(value), \\
& \quad \quad \text{untag}(\text{cdr}(\text{assoc}(a, bindings))) + index, \\
& \quad \quad \text{map-down-values}(mg\text{-}vars, bindings, temp\text{-}stk)))
\end{aligned}$$

EVENT: Disable mg-array-element-assignment-deposit-array-value-rewrite.

THEOREM: mg-array-element-assignment-step-25-no-error

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge (\neg \text{resources-inadequatep}(stmt, \\
& \quad \quad \text{proc-list}, \\
& \quad \quad \text{list}(\text{length}(temp\text{-}stk), \\
& \quad \quad \text{p-ctrl-stk-size}(ctrl\text{-}stk)))) \\
& \wedge (\text{car}(stmt) = \text{'predefined-proc-call-mg}) \\
& \wedge (\text{call-name}(stmt) = \text{'mg-array-element-assignment}) \\
& \wedge \text{ok-mg-statement}(stmt, r\text{-}cond\text{-}list, name\text{-}alist, proc\text{-}list) \\
& \wedge \text{ok-mg-def-plistp}(proc\text{-}list) \\
& \wedge \text{ok-mg-statep}(mg\text{-}state, r\text{-}cond\text{-}list) \\
& \wedge (\text{code}(\text{translate-def-body}(\text{assoc}(subr, proc\text{-}list), proc\text{-}list)) \\
& \quad = \text{append}(\text{code}(\text{translate}(cinfo, t\text{-}cond\text{-}list, stmt, proc\text{-}list)), \\
& \quad \quad code2))
\end{aligned}$$

```

 $\wedge$  user-defined-procp (subr, proc-list)
 $\wedge$  listp (ctrl-stk)
 $\wedge$  all-cars-unique (mg-alist (mg-state))
 $\wedge$  signatures-match (mg-alist (mg-state), name-alist)
 $\wedge$  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                     bindings (top (ctrl-stk)),
                                     temp-stk)
 $\wedge$  no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 $\wedge$  normal (mg-state)
 $\wedge$  ( $\neg$  negativep (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                mg-alist (mg-state)))))))
 $\wedge$  (idifference (caddr (call-actuals (stmt)),
                    untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                mg-alist (mg-state))))))  $\neq$  0))
 $\rightarrow$  (p-step (p-state (tag ('pc, cons (subr, length (code (cinfo)) + 7)),
                    ctrl-stk,
                    push (tag ('nat,
                                untag (mg-cond-to-p-nat (cc (mg-state),
                                                            t-cond-list)) - 1),
                                rput (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                mg-alist (mg-state))))),
                                untag (value (car (call-actuals (stmt)),
                                                bindings (top (ctrl-stk))))
                                + untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                mg-alist (mg-state))))),
                                map-down-values (mg-alist (mg-state),
                                                bindings (top (ctrl-stk)),
                                                temp-stk))),
                                translate-proc-list (proc-list),
                                list (list ('c-c,
                                            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                                MG-MAX-CTRL-STK-SIZE,
                                MG-MAX-TEMP-STK-SIZE,
                                MG-WORD-SIZE,
                                'run))
                    = p-state (tag ('pc,
                                    cons (subr,
                                        if normal (mg-meaning-r (stmt,
                                                                    proc-list,
                                                                    mg-state,
                                                                    n,
                                                                    list (length (temp-stk),
                                                                    p-ctrl-stk-size (ctrl-stk))))
                                        then length (code (translate (cinfo,

```

```

                                t-cond-list,
                                stmt,
                                proc-list)))
else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                proc-list,
                                                mg-state,
                                                n,
                                                list (length (temp-stk),
                                                            p-ctrl-stk-size (ctrl-stk))))),
                                label-alist (translate (cinfo,
                                                         t-cond-list,
                                                         stmt,
                                                         proc-list))),
                                append (code (translate (cinfo,
                                                         t-cond-list,
                                                         stmt,
                                                         proc-list)),
                                        code2)) endif)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (stmt,
                                          proc-list,
                                          mg-state,
                                          n,
                                          list (length (temp-stk),
                                                    p-ctrl-stk-size (ctrl-stk))))),
                bindings (top (ctrl-stk)),
                temp-stk),
translate-proc-list (proc-list),
list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-meaning-r (stmt,
                                                proc-list,
                                                mg-state,
                                                n,
                                                list (length (temp-stk),
                                                            p-ctrl-stk-size (ctrl-stk))))),
                                t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

THEOREM: mg-array-element-assignment-exact-time-lemma

$((n \neq 0)$
 $\wedge \quad (\neg \text{resources-inadequatep } (stmt,$

```

                                proc-list,
                                list (length (temp-stk),
                                        p-ctrl-stk-size (ctrl-stk))))
^ (car (stmt) = 'predefined-proc-call-mg)
^ (call-name (stmt) = 'mg-array-element-assignment)
^ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
^ ok-mg-def-plistp (proc-list)
^ ok-mg-statep (mg-state, r-cond-list)
^ (code (translate-def-body (assoc (subr, proc-list), proc-list))
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
              code2))
^ user-defined-procp (subr, proc-list)
^ listp (ctrl-stk)
^ all-cars-unique (mg-alist (mg-state))
^ signatures-match (mg-alist (mg-state), name-alist)
^ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                              bindings (top (ctrl-stk)),
                              temp-stk)
^ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
^ normal (mg-state))
→ (p (map-down (mg-state,
                proc-list,
                ctrl-stk,
                temp-stk,
                tag ('pc, cons (subr, length (code (cinfo))))),
        t-cond-list),
    clock (stmt, proc-list, mg-state, n))
= p-state (tag ('pc,
                cons (subr,
                    if normal (mg-meaning-r (stmt,
                                              proc-list,
                                              mg-state,
                                              n,
                                              list (length (temp-stk),
                                                  p-ctrl-stk-size (ctrl-stk))))
                    then length (code (translate (cinfo,
                                                  t-cond-list,
                                                  stmt,
                                                  proc-list)))
                    else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                                    proc-list,
                                                                    mg-state,
                                                                    n,
                                                                    list (length (temp-stk),

```

```

                                p-ctrl-stk-size (ctrl-stk))),
label-alist (translate (cinfo,
                        t-cond-list,
                        stmt,
                        proc-list)),
append (code (translate (cinfo,
                        t-cond-list,
                        stmt,
                        proc-list)),
code2)) endif)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (stmt,
                                         proc-list,
                                         mg-state,
                                         n,
                                         list (length (temp-stk),
                                                p-ctrl-stk-size (ctrl-stk)))),
bindings (top (ctrl-stk)),
temp-stk),
translate-proc-list (proc-list),
list (list ('c-c,
           mg-cond-to-p-nat (cc (mg-meaning-r (stmt,
                                                proc-list,
                                                mg-state,
                                                n,
                                                list (length (temp-stk),
                                                       p-ctrl-stk-size (ctrl-stk))))),
t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                 ;;
;;          EXACT-TIME-LEMMA for PREDEFINED's                      ;;
;;                                                                 ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

```

THEOREM: predefined-proc-call-exact-time-lemma
 $((n \neq 0)$
 $\wedge (\neg \text{resources-inadequatep (stmt,}$
 proc-list,

```

                                list (length (temp-stk),
                                      p-ctrl-stk-size (ctrl-stk))))
^  (car (stmt) = 'predefined-proc-call-mg)
^  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
^  ok-mg-def-plistp (proc-list)
^  ok-translation-parameters (cinfo, t-cond-list, stmt, proc-list, code2)
^  ok-mg-statep (mg-state, r-cond-list)
^  cond-subsetp (r-cond-list, t-cond-list)
^  (code (translate-def-body (assoc (subr, proc-list), proc-list))
      =  append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                code2))
^  user-defined-procp (subr, proc-list)
^  plistp (temp-stk)
^  listp (ctrl-stk)
^  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                              bindings (top (ctrl-stk)),
                              temp-stk)
^  no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
^  signatures-match (mg-alist (mg-state), name-alist)
^  normal (mg-state)
^  all-cars-unique (mg-alist (mg-state))
^  (¬ resource-errorp (mg-meaning-r (stmt,
                                     proc-list,
                                     mg-state,
                                     n,
                                     list (length (temp-stk),
                                             p-ctrl-stk-size (ctrl-stk)))))
→  (p (map-down (mg-state,
                proc-list,
                ctrl-stk,
                temp-stk,
                tag ('pc, cons (subr, length (code (cinfo)))),
                t-cond-list),
        clock (stmt, proc-list, mg-state, n))
    =  p-state (tag ('pc,
                    cons (subr,
                        if normal (mg-meaning-r (stmt,
                                                proc-list,
                                                mg-state,
                                                n,
                                                list (length (temp-stk),
                                                        p-ctrl-stk-size (ctrl-stk)))))
                    then length (code (translate (cinfo,
                                                    t-cond-list,

```

```

                                stmt,
                                proc-list)))
else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                proc-list,
                                                mg-state,
                                                n,
                                                list (length (temp-stk),
                                                            p-ctrl-stk-size (ctrl-stk))))),
                                label-alist (translate (cinfo,
                                                         t-cond-list,
                                                         stmt,
                                                         proc-list))),
                                append (code (translate (cinfo,
                                                         t-cond-list,
                                                         stmt,
                                                         proc-list))),
                                code2)) endif)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (stmt,
                                          proc-list,
                                          mg-state,
                                          n,
                                          list (length (temp-stk),
                                                    p-ctrl-stk-size (ctrl-stk))))),
                bindings (top (ctrl-stk)),
                temp-stk),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-meaning-r (stmt,
                                                proc-list,
                                                mg-state,
                                                n,
                                                list (length (temp-stk),
                                                            p-ctrl-stk-size (ctrl-stk))))),
                                t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```

EVENT: Make the library "**c-predefined4**".

Index

- add1-nat, 83
- all-cars-unique, 4–6, 8, 10, 12, 14, 16, 18, 20, 22, 25, 27, 30, 32, 35, 37, 38, 40, 43, 46, 47, 49, 51, 52, 55, 57, 59, 61, 63, 65, 68, 70, 72, 75, 78, 80, 81, 84, 85, 87, 89
- array-elemtype, 2, 45, 83
- array-identifierp, 1, 2, 45, 83, 84
- array-index-small-naturalp, 2
- array-length, 1–3, 45, 84
- arrays-have-non-zero-lengths, 1
- arrays-have-ok-values, 83
- bindings, 4–36, 38–44, 46–79, 81–83, 85–90
- boolean-identifierp, 1
- call-actuals, 2–36, 39, 41, 45–79, 82, 85
- call-name, 2, 3, 5, 6, 8, 10, 12, 14, 16, 18, 20, 22, 25, 27, 30, 32, 34, 36–38, 40, 43, 45–47, 49, 51, 52, 54, 57, 59, 61, 63, 65, 68, 70, 72, 75, 77, 79–81, 84, 87
- cc, 4–9, 11–17, 19–24, 26, 28, 29, 31–34, 36, 39–42, 44, 47, 48, 50, 52–54, 56, 58, 60, 62, 64, 65, 67, 69–74, 76, 77, 79, 82, 83, 85, 86, 88, 90
- character-identifierp, 1
- clock, 43, 87, 89
- code, 3–44, 46–90
- cond-subsetp, 89
- definedp, 3, 45
- deposit-array-value, 84
- deposit-temp, 83
- fetch-label, 39, 42, 44, 82, 86, 88, 90
- find-label, 39, 42, 44, 83, 86, 88, 90
- idifference, 1, 2, 14, 15, 17–20, 23–25, 27, 28, 30, 32, 35, 39, 41, 56–59, 61–63, 66, 68, 70, 73, 75, 76, 78, 82, 85
- idifference-lessp, 2
- idifference-lessp2, 1
- int-identifierp, 1, 2, 45
- integerp, 1, 24
- label-alist, 39, 42, 44, 82, 86, 88, 90
- length, 2–44, 46–90
- lessp-plus-transitive, 1
- limits-for-small-integerp, 2
- map-down, 4, 43, 46, 87, 89
- map-down-values, 4–9, 11–17, 19–24, 26, 28, 29, 31–36, 39–42, 44, 47–50, 52–56, 58, 60, 62, 64–67, 69–74, 76, 77, 79, 82–86, 88, 90
- maxint, 2
- mg-alist, 2–83, 85–90
- mg-alistp, 1, 2, 83, 84
- mg-array-element-assignment-arg
 - 3-simple, 45
 - 4-small-integerp, 45
 - s-definedp, 45
 - s-have-simple-mg-type-refps, 44
- mg-array-element-assignment-deposit-array-value-rewrite, 84
- mg-array-element-assignment-exact-time-lemma, 86
- mg-array-element-assignment-index-lessp-temp-stk-length, 75
- mg-array-element-assignment-last-step-error-case, 81
- mg-array-element-assignment-push-cc, 79
- mg-array-element-assignment-step

- 12-no-error, 54
- 13-index-error, 56
- 13-no-error, 60
- 14-no-error, 63
- 15-no-error, 65
- 18-no-error, 70
- 19-no-error, 72
- 20-no-error, 75
- 25-no-error, 84
- 5, 47
- s-1-4, 46
- s-14-16-index-error, 59
- s-16-17-no-error, 67
- s-21-22-no-error, 77
- s-6-8, 48
- s-9-11-no-error, 52
- s-9-12-neg-index, 50
- mg-array-element-assignment-sub
1-cc, 80
- mg-cond-to-p-nat, 4–9, 11–17, 19–
24, 26, 28, 29, 31–34, 36,
40–42, 44, 47, 48, 50, 52–
54, 56, 58, 60, 62, 64, 65,
67, 69–74, 76, 77, 79, 83,
85, 86, 88, 90
- mg-index-array-arg4-small-intege
rp, 3
- mg-index-array-args-definedp, 3
- mg-index-array-args-have-simple
-mg-type-refps, 2
- mg-index-array-exact-time-lemma, 42
- mg-index-array-index-lessp-temp
-stk-length, 27
- mg-index-array-last-step-error-
case, 38
- mg-index-array-push-cc, 36
- mg-index-array-step-12-no-error, 12
- mg-index-array-step-13-index-er
ror, 14
- mg-index-array-step-13-no-error, 18
- mg-index-array-step-16-no-error, 22
- mg-index-array-step-17-no-error, 24
- mg-index-array-step-18-no-error, 27
- mg-index-array-step-19-no-error, 29
- mg-index-array-step-20-no-error, 32
- mg-index-array-step-25-no-error, 40
- mg-index-array-step-5, 4
- mg-index-array-steps-1-4, 3
- mg-index-array-steps-14-15-no-e
rror, 20
- mg-index-array-steps-14-16-inde
x-error, 16
- mg-index-array-steps-21-22-no-e
rror, 34
- mg-index-array-steps-6-8, 6
- mg-index-array-steps-9-11-no-er
ror, 10
- mg-index-array-steps-9-12-neg-i
ndex, 8
- mg-index-array-sub1-cc, 37
- mg-max-ctrl-stk-size, 4–9, 11–24, 26,
28, 29, 31–34, 36–42, 44,
47, 48, 50, 52–54, 56, 58–
60, 62, 64, 65, 67, 69–72,
74, 76, 77, 79–83, 85, 86,
88, 90
- mg-max-temp-stk-size, 2, 4–24, 26–
29, 31–34, 36–42, 44, 47,
48, 50, 52–54, 56, 58–60,
62, 64, 65, 67, 69–72, 74,
76, 77, 79–83, 85, 86, 88,
90
- mg-meaning-r, 39–44, 82, 83, 85–90
- mg-to-p-simple-literal, 8–36, 39, 41,
50–79, 82–85
- mg-var-ok-array-index-ok3, 2
- mg-vars-list-ok-in-p-state, 2, 4–6, 8,
10, 12, 14, 16, 18, 20, 22,
25, 27, 28, 30, 32, 35, 38,
41, 43, 46, 47, 49, 51, 53,
55, 57, 59, 61, 63, 65, 68,
70, 73, 75, 78, 81, 84, 85,
87, 89
- mg-word-size, 4–24, 26–29, 31–34,
36–42, 44, 47, 48, 50, 52–
54, 56, 58–60, 62, 64, 65,
67, 69–72, 74, 76, 77, 79–
83, 85, 86, 88, 90

nat-p-objectp-reduction, 2
 no-p-aliasing, 4–6, 8, 10, 12, 14, 17,
 18, 20, 22, 25, 28, 30, 32,
 35, 38, 41, 43, 46, 47, 49,
 51, 53, 55, 57, 59, 61, 63,
 65, 68, 70, 73, 75, 78, 82,
 84, 85, 87, 89
 non-negative-integerp-small-nat
 uralp, 24
 normal, 4–6, 8, 10, 12, 14, 17, 18,
 20, 22, 25, 28, 30, 32, 35,
 37–39, 41, 43, 46, 47, 49,
 51, 53, 55, 57, 59, 61, 63,
 65, 68, 70, 73, 75, 78, 80–
 82, 85, 87, 89
 not-zero-p-mg-array-element-assi
 gnment-arg4, 45
 not-zero-p-mg-index-array-arg4, 3

 ok-mg-array-value, 83
 ok-mg-def-plistp, 3, 5, 6, 8, 10, 12,
 14, 16, 18, 20, 22, 25, 27,
 30, 32, 34, 36–38, 40, 43,
 46, 47, 49, 51, 52, 55, 57,
 59, 61, 63, 65, 68, 70, 72,
 75, 78–81, 84, 87, 89
 ok-mg-statement, 2, 3, 5, 6, 8, 10,
 12, 14, 16, 18, 20, 22, 25,
 27, 30, 32, 34, 36–38, 40,
 43, 45–47, 49, 51, 52, 54,
 57, 59, 61, 63, 65, 68, 70,
 72, 75, 77, 79–81, 84, 87,
 89
 ok-mg-statep, 2, 3, 5, 6, 8, 10, 12,
 14, 16, 18, 20, 22, 25, 27,
 30, 32, 34, 36–38, 40, 43,
 45–47, 49, 51, 52, 55, 57,
 59, 61, 63, 65, 68, 70, 72,
 75, 78, 80, 81, 84, 87, 89
 ok-translation-parameters, 89

 p, 43, 87, 89
 p-ctrl-stk-size, 3, 4, 6, 8, 10, 12, 14,
 16, 18, 20, 22, 24, 27, 29,
 32, 34, 36–44, 46, 47, 49,
 51, 52, 54, 57, 59, 60, 63,
 65, 68, 70, 72, 75, 77, 79–
 90
 p-frame, 6–9, 11, 13–17, 19, 21–26,
 28–31, 33–35, 48–50, 52–
 56, 58, 60–62, 64, 66, 67,
 69, 71–74, 76–78
 p-object-type-int, 1
 p-objectp-type, 1, 2
 p-state, 4–24, 26–29, 31–34, 36–42,
 44, 47, 48, 50, 52–54, 56,
 58–60, 62, 64, 65, 67, 69–
 72, 74, 76, 77, 79–83, 85,
 86, 88, 90
 p-step, 4, 5, 7, 9, 11, 13, 15, 17, 19,
 21, 23, 26, 28, 31, 33, 36–
 39, 41, 46, 48, 50, 52, 53,
 56, 58, 60, 62, 64, 67, 69,
 71, 74, 76, 79–82, 85
 p-word-size, 1, 2
 plistp, 89
 predefined-proc-call-exact-time
 -lemma, 88
 push, 4–9, 11–17, 19–26, 28–35, 37–
 39, 41, 47–50, 52–56, 58,
 60–62, 64–67, 69–74, 76–
 78, 80–82, 85
 put, 83, 84
 put-deposit-array-value-inducti
 on-hint, 83
 put-deposit-array-value-rewrite, 84
 put-preserves-ok-mg-array-value, 83

 resource-errorp, 89
 resources-inadequatep, 3, 4, 6, 8, 10,
 12, 14, 16, 18, 20, 22, 24,
 27, 29, 32, 34, 36–38, 40,
 43, 46, 47, 49, 51, 52, 54,
 57, 59, 60, 63, 65, 68, 70,
 72, 75, 77, 79–81, 84, 87,
 89
 rget, 29, 31, 33–36, 41
 rput, 34, 36, 41, 77, 79, 84, 85

signatures-match, 2–6, 8, 10, 12, 14,
 16, 18, 20, 22, 25, 27, 30,
 32, 35, 37, 38, 41, 43, 45–
 47, 49, 51, 53, 55, 57, 59,
 61, 63, 65, 68, 70, 72, 75,
 78, 80, 81, 85, 87, 89
 simple-identifierp, 1, 2, 45
 simple-identifierp-options, 1
 simple-typed-identifier-simple-i-
 dentfierp, 2
 simple-typed-identifier-type-eq
 uivalence, 40
 simple-typed-identifierp, 2, 40, 45
 simple-typed-literalp, 83
 small-integerp, 1–3, 24, 45
 small-integerp-difference, 1
 small-naturalp, 2, 24

 tag, 1, 2, 4–26, 28–39, 41–44, 46–74,
 76–83, 85–90
 top, 4–36, 38–44, 46–79, 81–83, 85–
 90
 translate, 3, 5, 6, 8, 10, 12, 14, 16,
 18, 20, 22, 25, 27, 30, 32,
 34, 37–40, 42–44, 46, 47,
 49, 51, 52, 55, 57, 59, 61,
 63, 65, 68, 70, 72, 75, 78,
 80–84, 86–90
 translate-def-body, 3, 5, 6, 8, 10, 12,
 14, 16, 18, 20, 22, 25, 27,
 30, 32, 34, 36–38, 40, 43,
 46, 47, 49, 51, 52, 55, 57,
 59, 61, 63, 65, 68, 70, 72,
 75, 78, 80, 81, 84, 87, 89
 translate-proc-list, 4–9, 11–24, 26,
 28, 29, 31–34, 36–42, 44,
 47, 48, 50, 52–56, 58, 60,
 62, 64–67, 69–74, 76, 77,
 79–83, 85, 86, 88, 90

 untag, 2, 9, 10, 12, 14, 15, 17–20,
 22–36, 38, 39, 41, 51, 53,
 55–59, 61–63, 65, 66, 68,
 70, 72–79, 81, 82, 84, 85

 user-defined-procp, 4–6, 8, 10, 12,
 14, 16, 18, 20, 22, 25, 27,
 30, 32, 34, 37, 38, 40, 43,
 46, 47, 49, 51, 52, 55, 57,
 59, 61, 63, 65, 68, 70, 72,
 75, 78, 80, 81, 85, 87, 89

 value, 2, 4–7, 9–13, 15–36, 41, 46–
 51, 53–79, 85

 zerop-integerp-trichotomy, 1