A Suite of Hard ACL2 Theorems Arising in Refinement-Based Processor Verification

Panagiotis (Pete) Manolios
Sudarshan Srinivasan

Georgia Institute of Technology

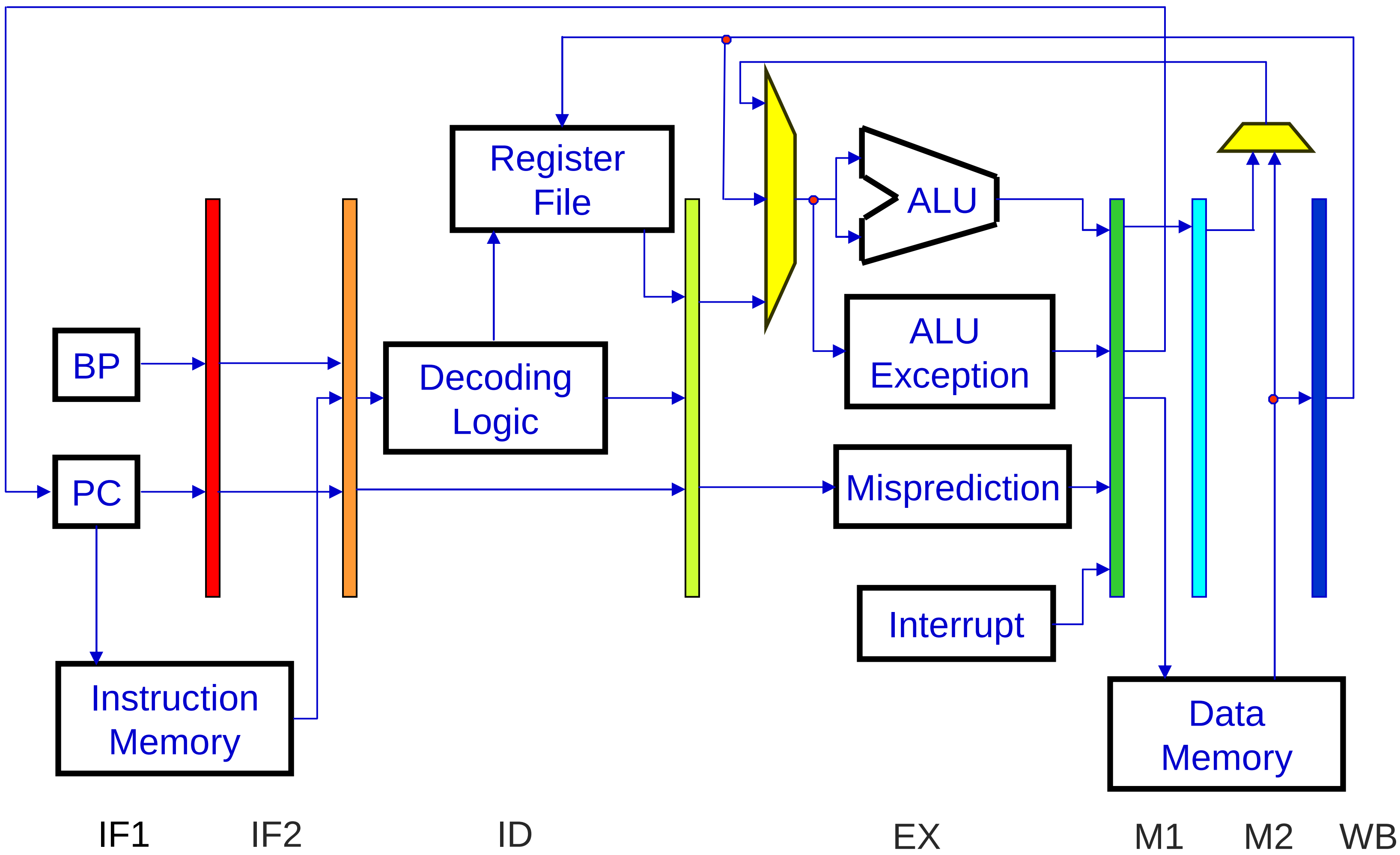
Introduction

- Hardware verification is an area of strength for ACL2.
  - Efficiently executable microprocessor models.
  - Various levels of abstraction, including bit- & cycle-accurate.
  - Floating point verification.
- We identify a class of “naturally arising” hardware verification problems that are hard for ACL2.
- *But*, other tools (UCLID) easily handle the problems.
- Our goal is to stimulate research on improving ACL2.
- We propose an approach on integrating decision procedures and want feedback.

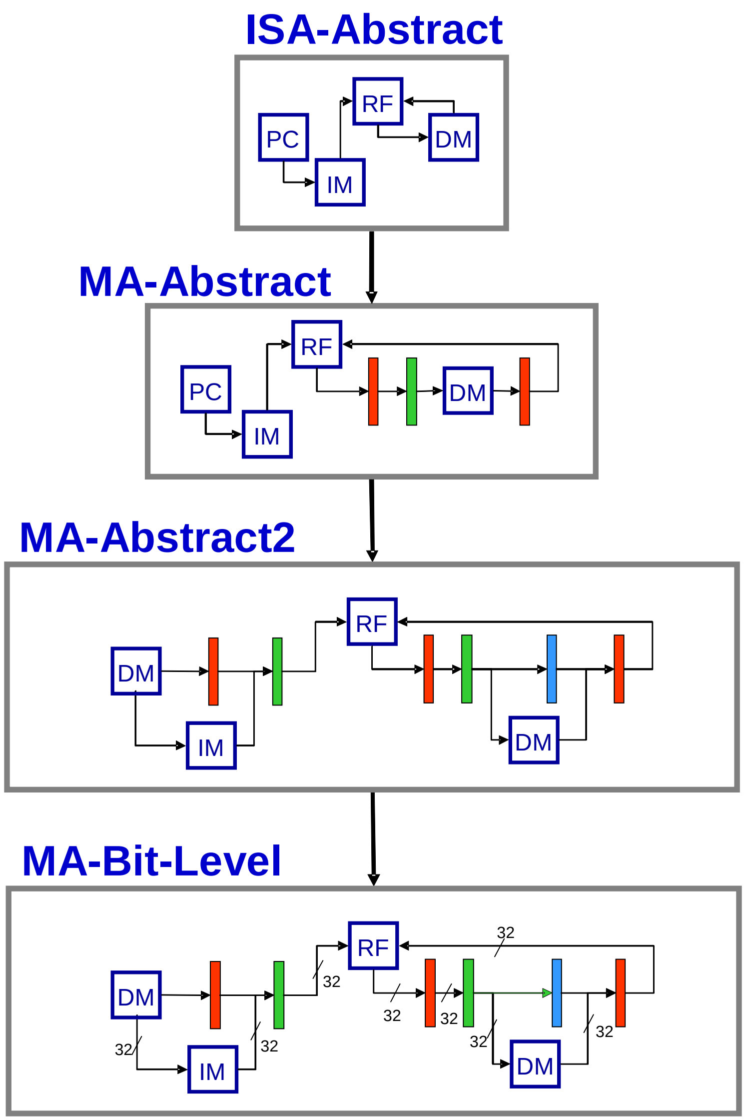
Outline

- Processor Models.
- Refinement.
- Refinement in ACL2.
- UCLID System.
- Results.
- Integrating UCLID with ACL2.
- Conclusions and Future Work.

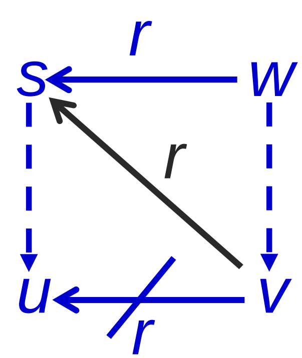
Processor Model



Refinement, the Picture

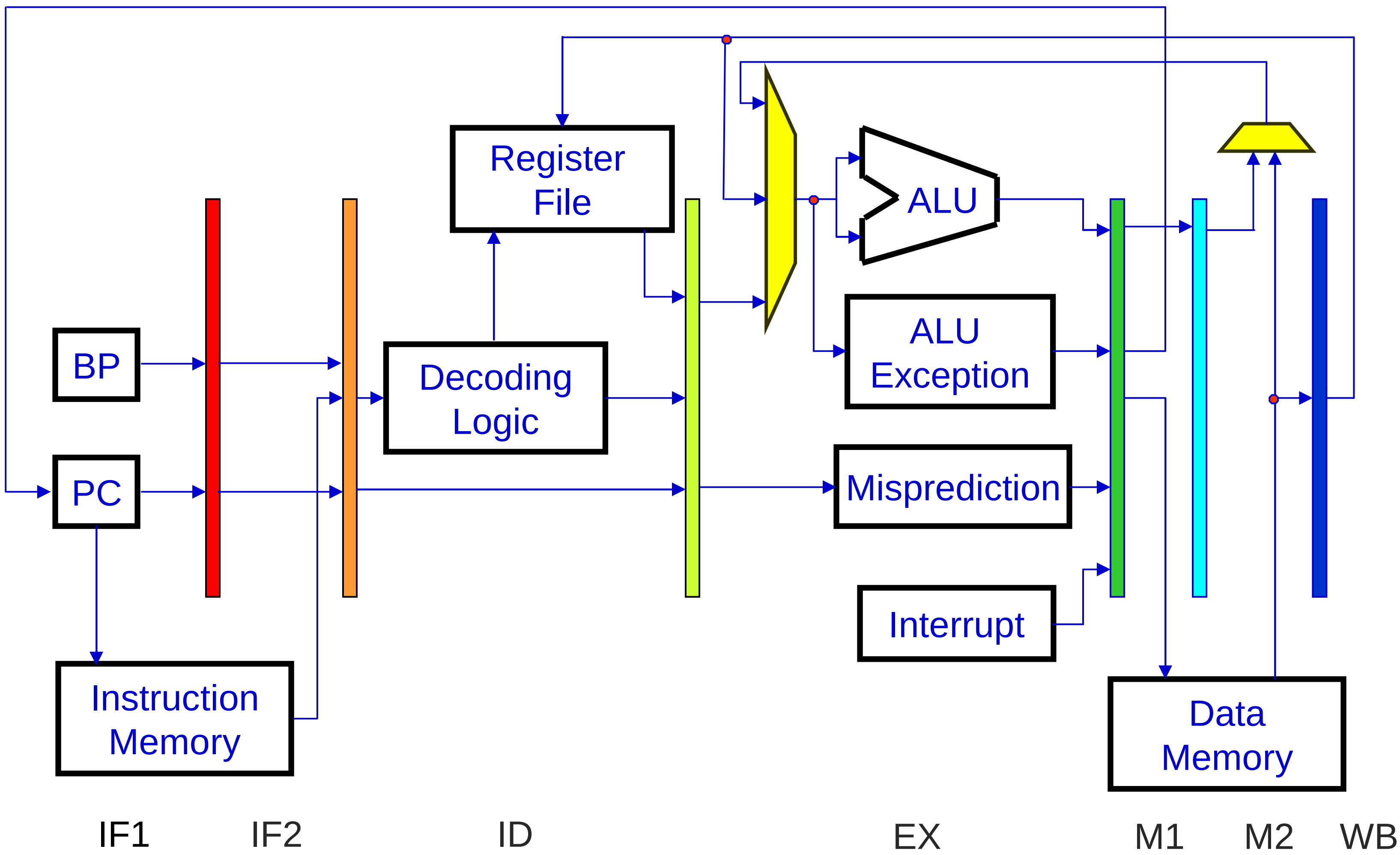


- Formal connection between different abstraction levels.
- Compositional.
- Avoid “Leaky Abstractions.”

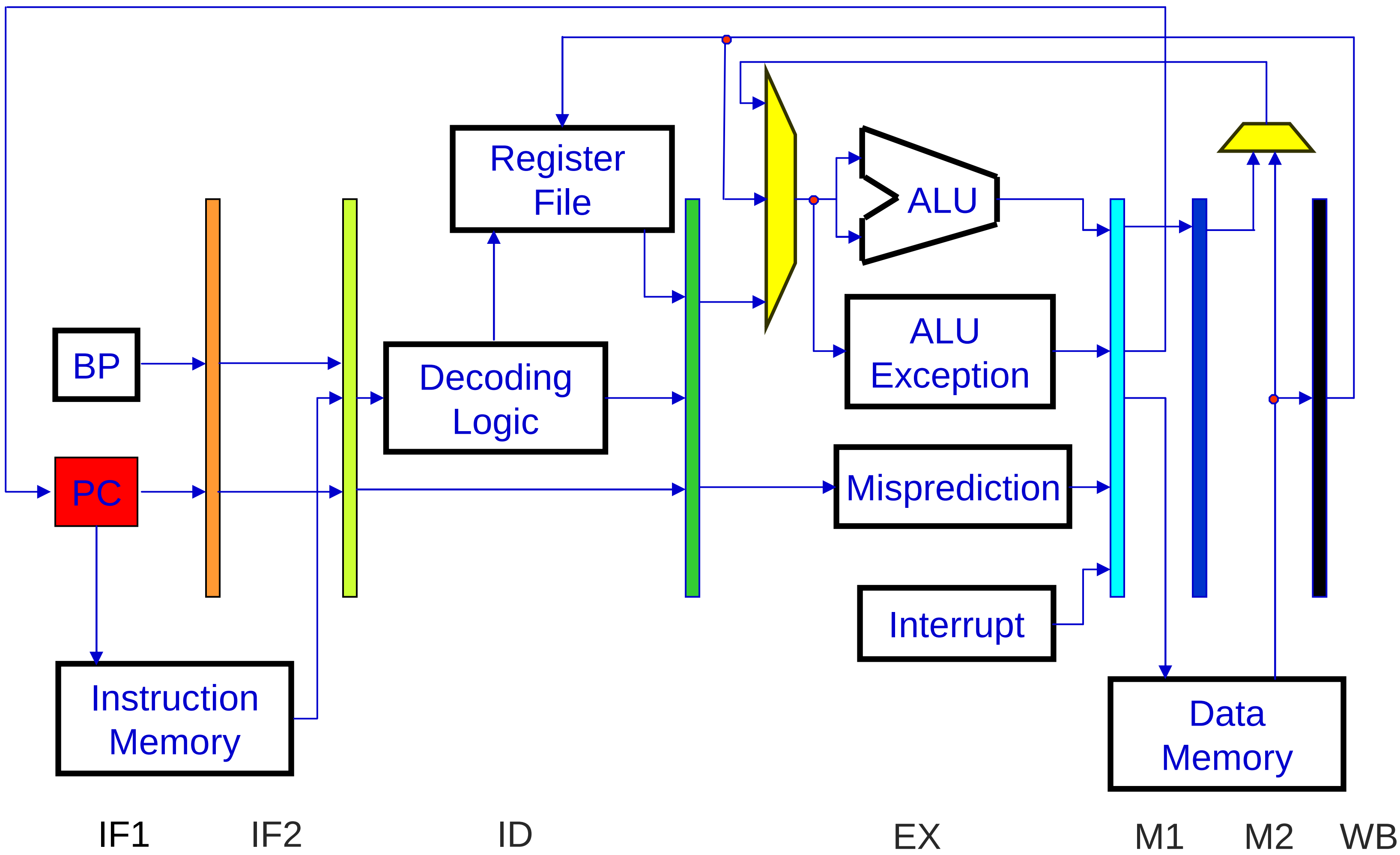


rank.v < rank.w

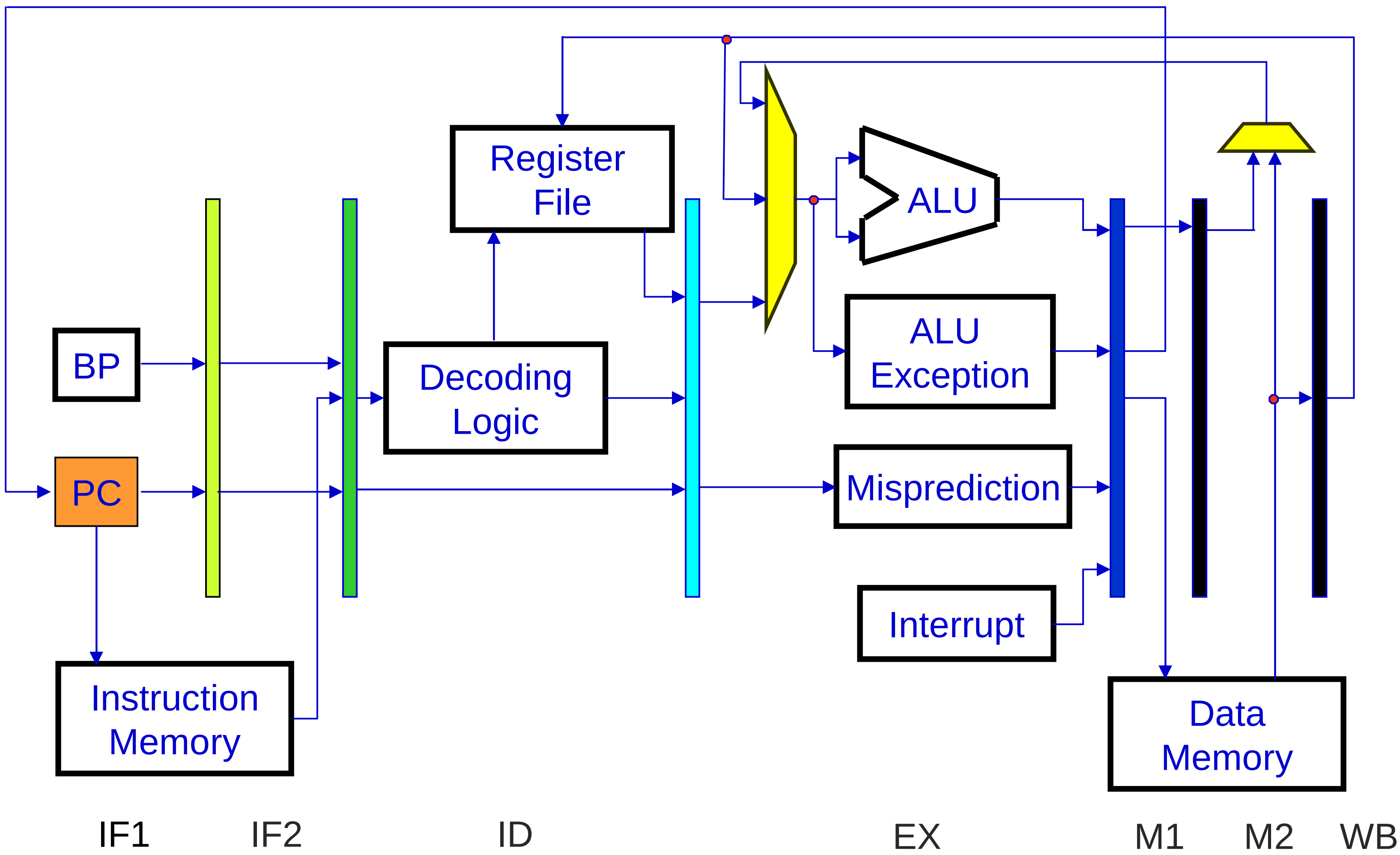
Processor Model: Commitment



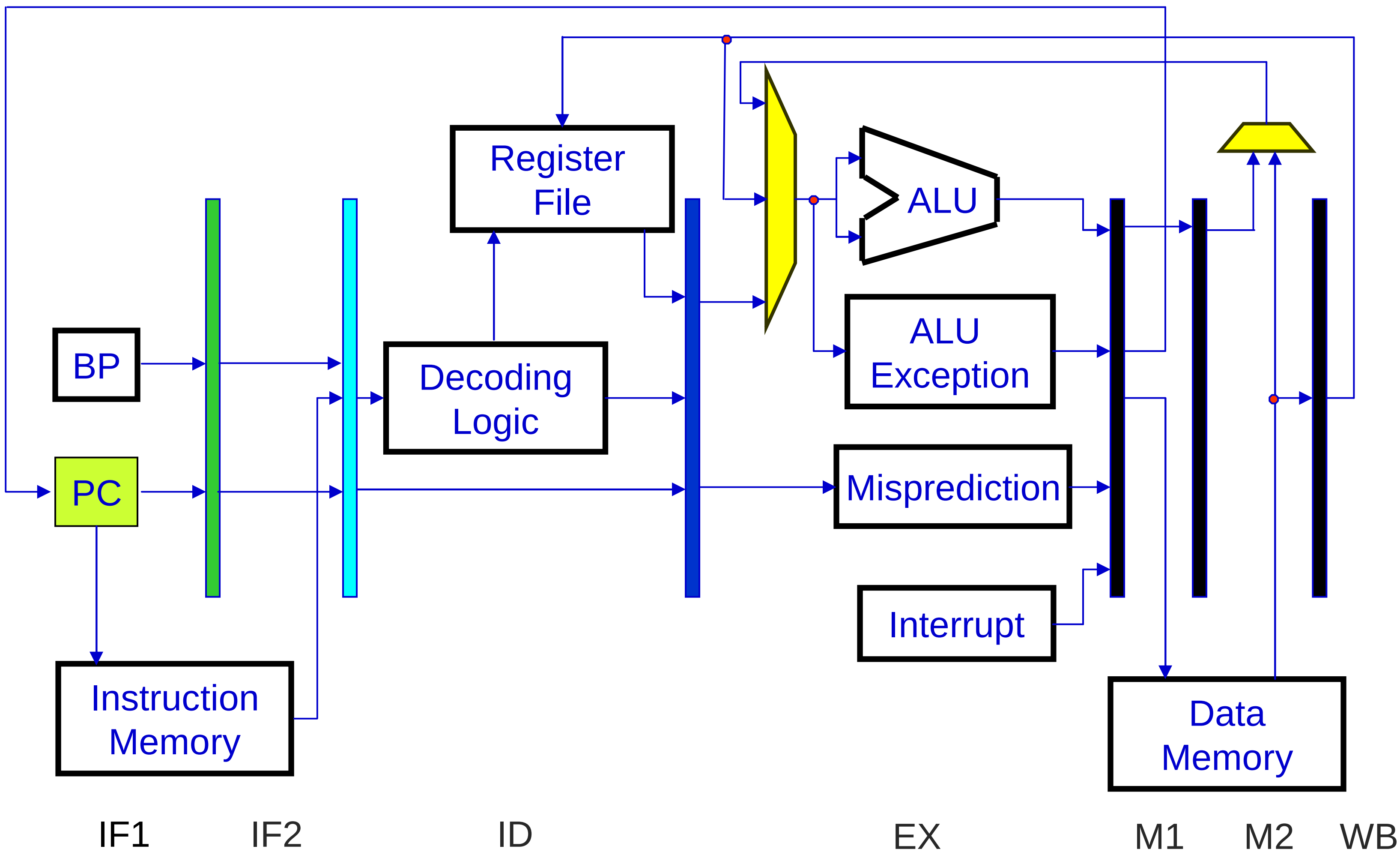
Processor Model: Commitment



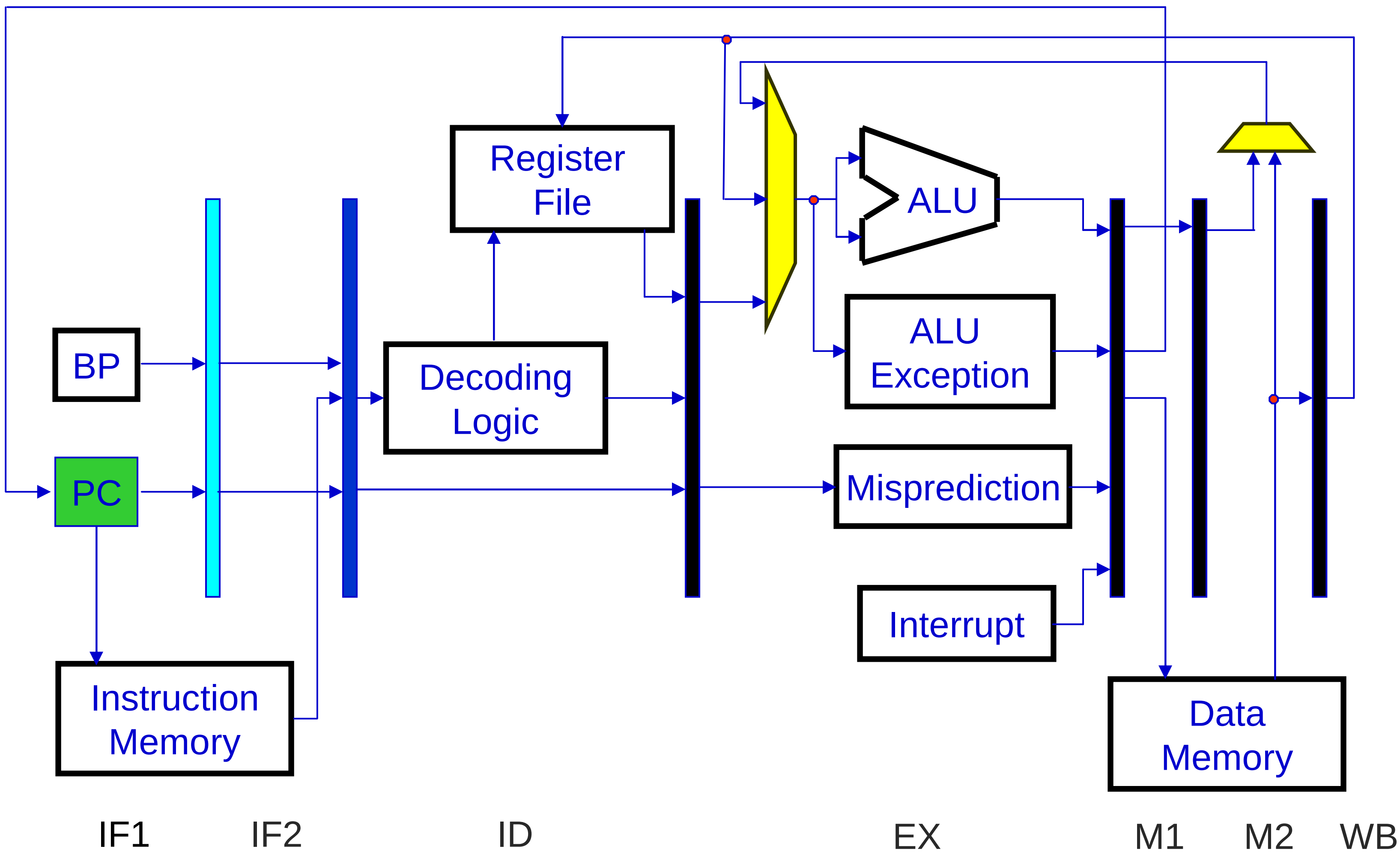
Processor Model: Commitment



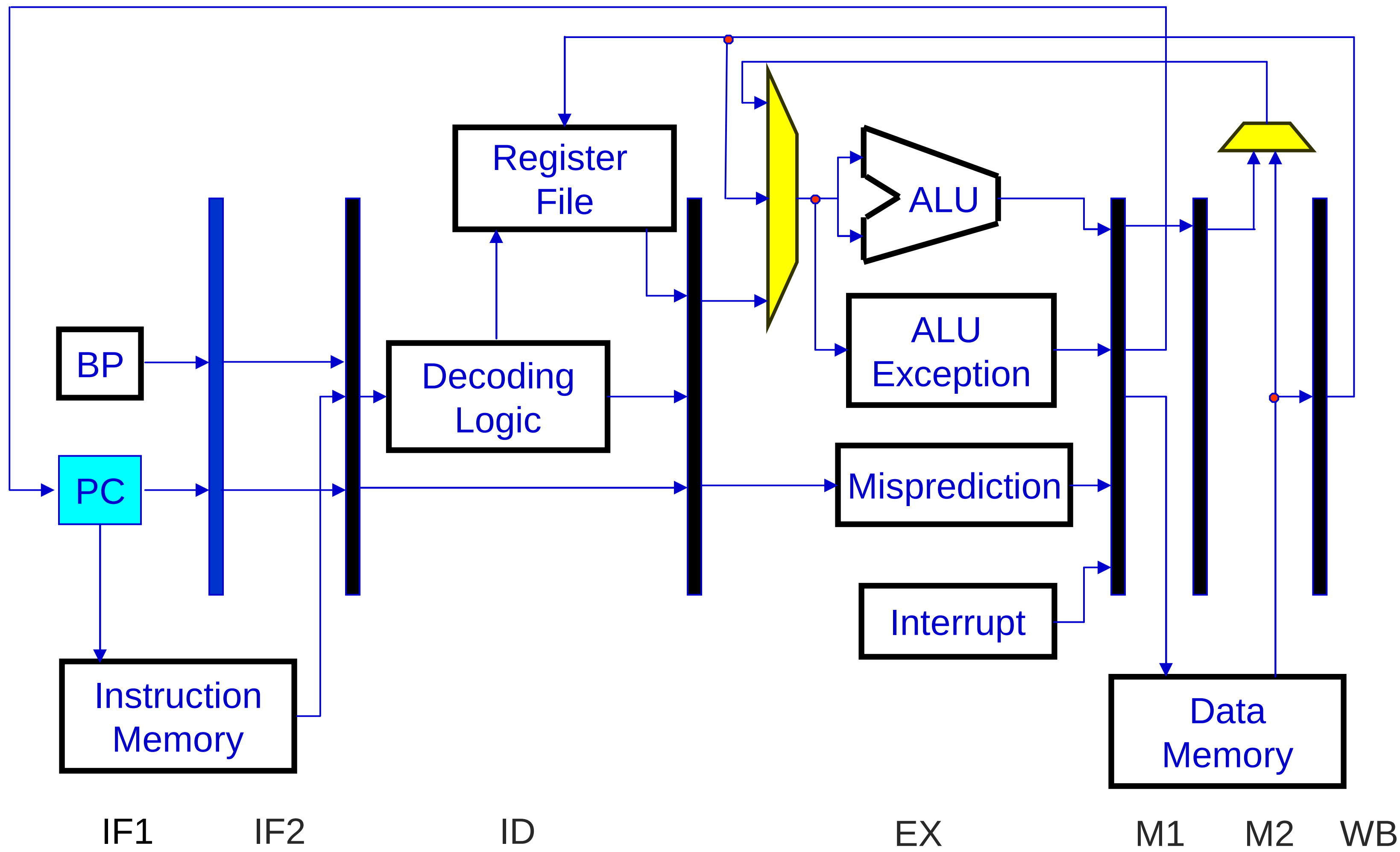
Processor Model: Commitment



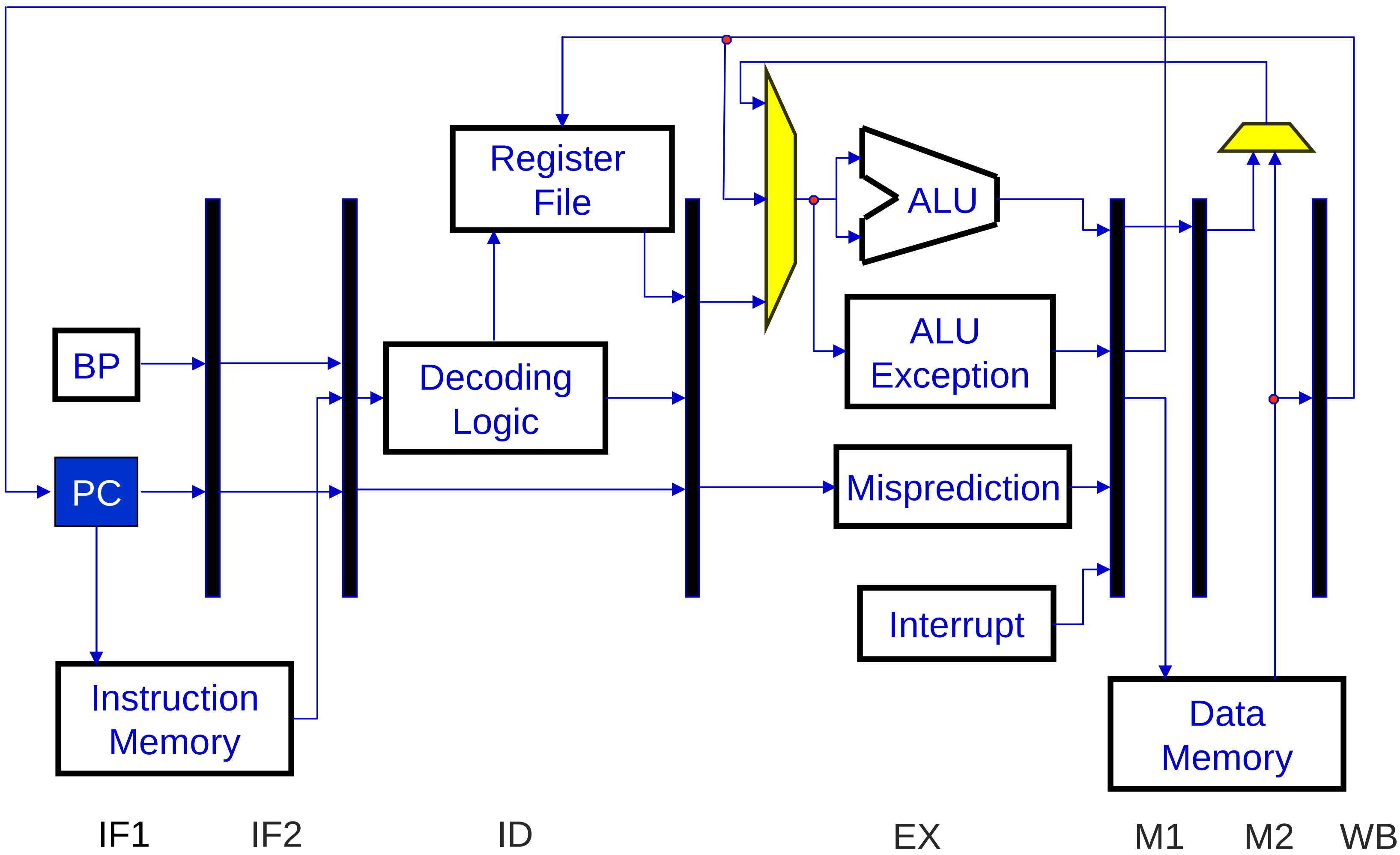
Processor Model: Commitment



Processor Model: Commitment



Processor Model: Commitment



Refinement Maps

- Commitment.
  - Partially executed instructions are invalidated.
  - Roll back the MA to the last committed instruction.
  - Requires an invariant that characterizes the reachable states that we call the "Good MA" invariant.
- Flushing.
  - Dual of commitment, partially executed instructions are flushed.
  - Safety proof for our examples similar to Burch and Dill notion of correctness.
  - No invariant required.
- Refinement maps and the Good MA invariant are implemented by stepping the processor model.

Refinement Theorems in ACL2

```
(defthm WEB_CORE
  (implies
   (and
    (integerp fdpPC0)
    (integerp depPC0)
    (booleanp deRegWrite0)
    …)
   (let* ((ST0 (initialize fdpPC0 depPC0 ...))
          (ST1 (simulate ST0 nil pc0 nil nil pc0
      ..))
    ...
          (Good_MA_V  (Good_MA_a
                       Equiv_MA_0
                       Equiv_MA_1
                       Equiv_MA_2
                       Equiv_MA_3
                       Equiv_MA_4))
   …
```

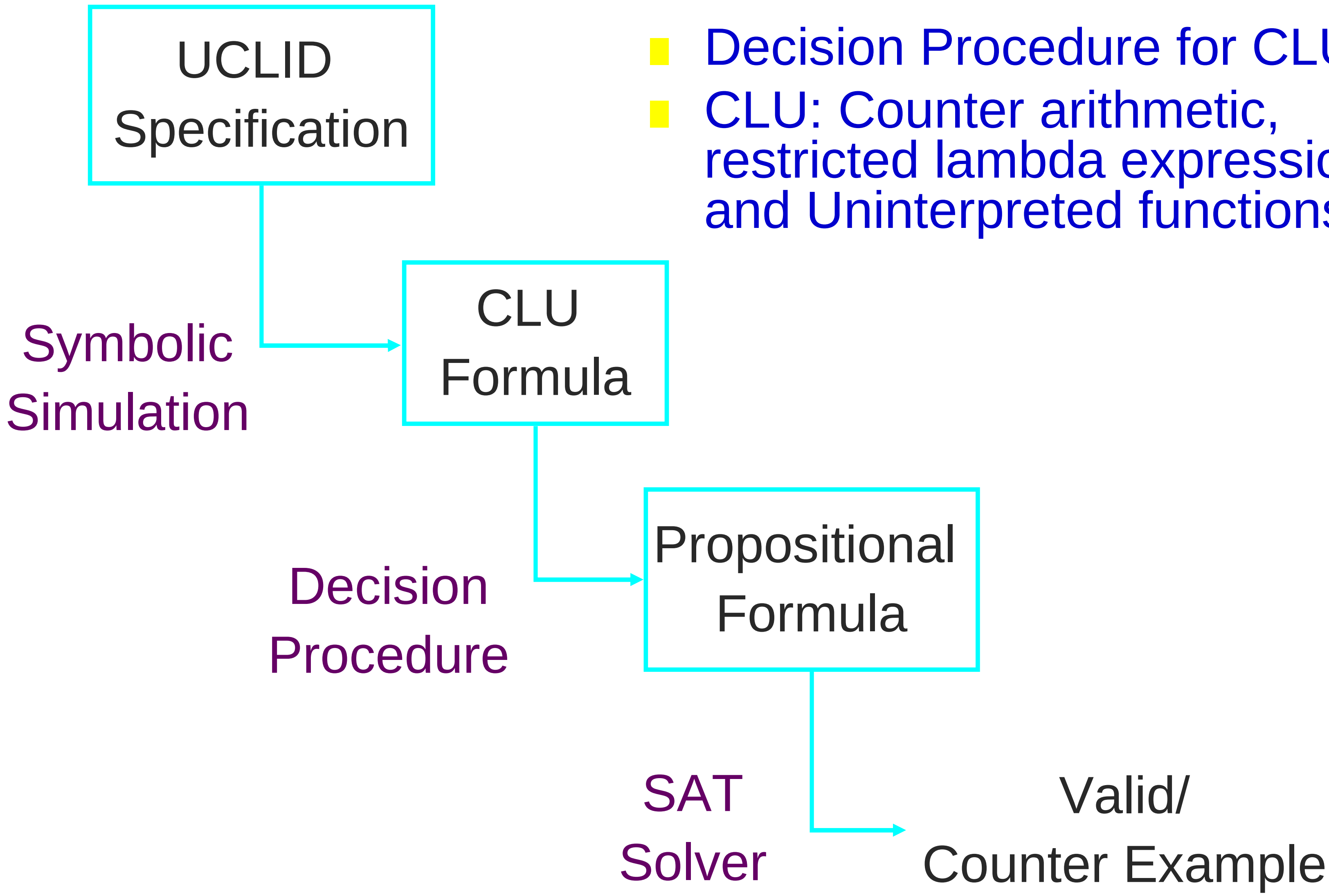
```
     (Rank_V (rank_a
                 (g 'mwWRT (g 'impl ST34))
                  (g 'emWRT (g 'impl ST34))
                  (g 'deWRT (g 'impl ST34))
                  (g 'fdWRT (g 'impl ST34))
                  ZERO))
     (S_pc1 (g 'sPC (g 'speci ST35)))
     (S_rf1 (g 'sRF (g 'speci ST35)))
     (S_dmem1 (g 'sDMem (g 'speci ST35))))
  (and
   Good_MA_V
   (or
    (not
     (and
      (equal S_pc0 I_pc0)
       (equal S_dmem0 I_dmem0))) …)
```

Refinement Theorems in ACL2

- Historical perspective.
  - Considerable effort expended in automating refinement in ACL2.
  - Even so, refinement proofs of simple machines took >1,000 secs.
  - E.g., correctness of 5 stage pipeline (translated from UCLID) took 15.5 days for ACL2 to prove.
  - UCLID took 3 secs to prove the same theorem!
- Our suite consists of refinement theorems translated from UCLID specifications.
- While far from perfect, the translator is reasonable.
  - Model written for ACL2: 130 secs.
  - Model translated from UCLID: 430 secs.

UCLID System

- Decision Procedure for CLU.
- CLU: Counter arithmetic, restricted lambda expressions, and Uninterpreted functions.



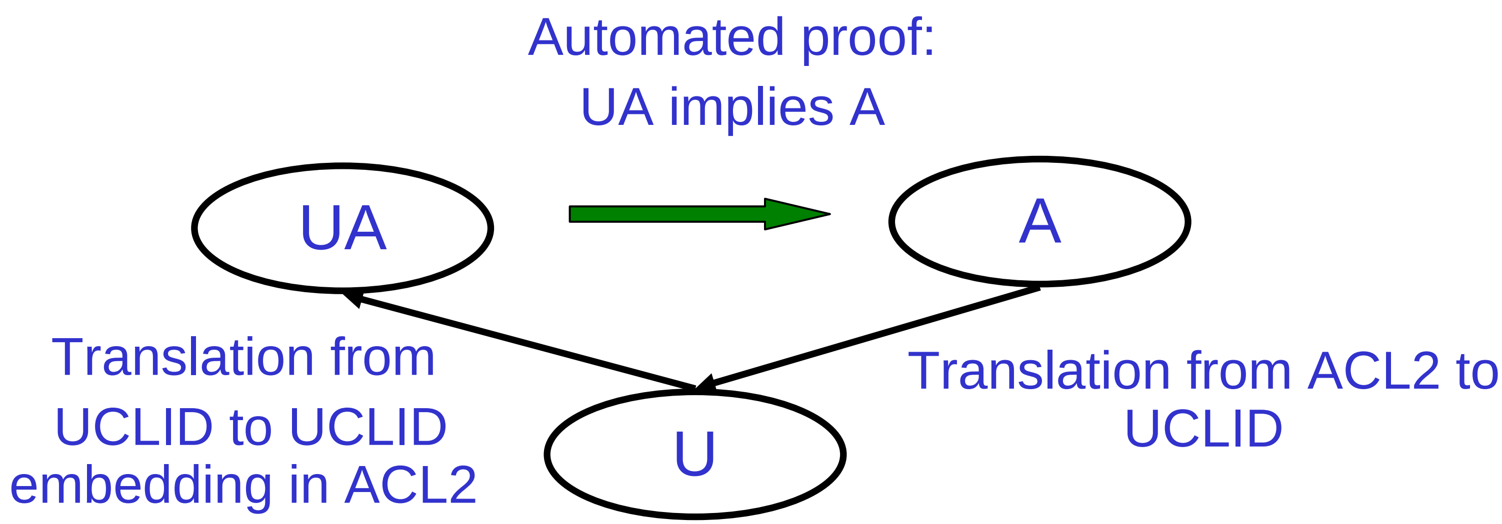
Theorems and Results

Theorems	CNF Vars	CNF Clauses	UCLID [sec]			ACL2 [sec]
			UCLID	Siege	Total	
5S-Part	5,285	15,457	1	2	3	1,339,200
5S-SL	5,285	15,457	1	2	3	*1,339,200*
CXS-SL	12,495	36,925	3	29	32	*14,284,800*
CXS-BP-SL	23,913	70,693	5	300	305	*136,152,000*
CXS-BP-EX-SL	24,149	71,350	5	233	238	*106,243,200*
CXS-BP-EX-INP-SL	24,478	72,322	6	263	269	*120,081,600*
FXS-SL	53,441	159,010	15	160	175	*78,120,000*
FXS-BP-SL	71,184	211,723	16	187	203	*90,619,200*
FXS-BP-EX-SL	74,591	221,812	17	163	180	*80,352,000*
FXS-BP-EX-INP-SL	81,121	241,345	19	170	189	*84,369,600*

Integrating UCLID with ACL2

- Core refinement theorem is CLU expressible.
- Limitations of UCLID:
  - Abstract models.
  - Models not executable.
  - We ultimately want bit-level verification.
  - Restricted logic and specification language.
    - Polluted models.
    - Full refinement theorem not expressible.
- Our approach: coarse grained integration.

Integrating UCLID with ACL2



A : ACL2 theorem

U : UCLID formula

UA : Translation of U, using the embedding of UCLID in ACL2

Conclusions and Future Work

- Presented a class of “naturally occurring” problems that ACL2 has difficulty handling.
- We hope to stimulate research in improving ACL2.
- Future work: Integrating decision procedures (UCLID) with ACL2.