CS 429 Quiz 3: October 30, 2017

Note that this quiz has two sides.

1. (5 points) For each of the following, circle A if it’s an architectural feature of the ISA and I if it’s an implementation feature.

   (a) A I Number of general purpose registers.
   (b) A I Number of cycles to execute an integer multiply.
   (c) A I Format for floats.
   (d) A I Average speed of memory access.
   (e) A I Condition codes set by ADD instruction.

2. (2 points) On an x86-64 Linux system, which of these take up the most bytes in memory?
   A. char a[7]
   B. float d
   C. short b[3]
   D. int *c

3. (2 points) What is the C equivalent of movq 0x10(%rax, %rcx, 4), %rdx?
   A. rdx = *(rax + rcx*4 + 0x10)
   B. rdx = *(rax + rcx + 4 + 0x10)
   C. rdx = rax + rcx + 4 + 10
   D. *(rax + rcx + 4 + 10) = rdx

4. (2 points) Which of the following x86 instructions can be used to add two registers and store the result without overwriting either of the original registers?
   A. movq B. leaq C. addq D. cmove E. none of these
5. (2 points) Given the declaration `int A[5][7];` the memory address `A[r][c]` can be calculated as

A. `A + 28 * r + 4 * c`
B. `A + 5 * r + 7 * c`
C. `A + 20 * (r-1) + 28 * (c-1)`
D. `A + 20 * r + 28 * c`
E. None of the above.

6. (2 points) Consider the following datatype definition on an x86-64 machine:

```c
typedef struct {
    double *p;
    char c1;
    int i;
    char c2;
} struct1;
```

How many bytes are allocated for an object of type `struct1`?
A. 4 B. 8 C. 16 D. 20 E. 24 F. None of the above

7. (2 points) Consider the following snippet of x86-64 code:

```
400544: callq 400580
40054D: movq %rax, (%rbx)
```

with the following values in locations:

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>%rsp</td>
<td>0x120</td>
</tr>
<tr>
<td>%rip</td>
<td>0x400544</td>
</tr>
</tbody>
</table>

What will be the values respectively of `%rsp`, `%rip` and the top of the stack following the next instruction execution?

A. 0x128, 0x400580, 0x40054D
B. 0x118, 0x400580, 0x40054D
C. 0x128, 0x40054D, 0x400580
D. 0x118, 0x40054D, 0x400580
E. none of these