CS 3330 Exam 1 Fall 2019

Name: EXAM KEY **Computing ID:** KEY

Letters go in the boxes unless otherwise specified (e.g., for **C** 8 write "C" not "8").
Write Letters clearly: if we are unsure of what you wrote you will get a zero on that problem.
Bubble and Pledge the exam or you will lose points.
Assume unless otherwise specified:

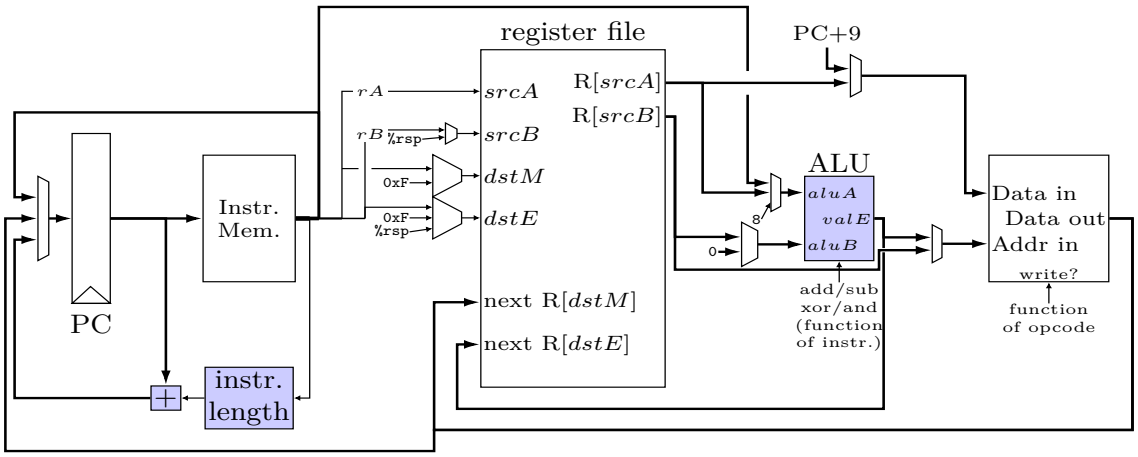
- little-endian 64-bit architecture
- `%rsp` points to the most recently pushed value, not to the next unused stack address.
- questions are single-selection unless identified as select-all

Variable Weight: point values per question are marked in square brackets.
Mark clarifications: If you need to clarify an answer, do so, and also add a ⋆ to the top right corner of your answer box.

. . . . . .

Information for questions 1–3

Suppose we wanted to add a `rrswap rA, rB` instruction to the single-cycle Y86 processor design shown below:



This instruction would take two registers and swap their values. For example, if `%rax` initially contained `0x1234` and `%rbx` initially conained `0x5678`,then running `rrswap %rax, %rbx` would result in `%rbx` containing `0x1234` and `%rax` containing `0x5678`.

Question 1 [2 pt]: (see above) The encoding for the `rrswap` instruction would probably be most similar to the encoding of _____.

A `popq`
B `irmovq`
C `addq`
D `pushq`

Answer: C

Question 2 [2 pt]: (see above) Which of the following changes would be helpful for implementing the instruction on the processor design shown above? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [✓] adding a MUX just before the register file's next R[dstM] input

B [ ] adjusting the MUXes controlling the ALU inputs to allow **either** inputs to be set to R[srcA]'s value

C [ ] adjusting the MUX controlling the `srcA` input of the register file to add an additional input

D [ ] adding an additional read and/or write port to the register file

Question 3 [2 pt]: (see above) When the `rrswap` instruction is executing, what should the `srcB` input to the register file select?

A the value `rB` from the instruction
B another value (an additional input needs to be added to the MUX)
C the constant register number for `%rsp`
D it does not matter

Answer: A

Question 4 [2 pt]: Consider a machine with 4 condition codes OF (overflow flag), SF(sign flag), Carry flag (CF), Zero flag. What flag(s) is/are set after the code executes? Assume all registers (including the flag registers) originally contain 0.

```
addq $1, %rbx
subq $0xFFFFFFFD, %rcx
```

Place a ✓in each box corresponding to a correct answer and leave other boxes blank.

A ☐ SF

B ☐ OF

C ☐ ZF

D ☐ CF

Also Accepted SF & CF
if you interpreted the numbers as unsigned

Information for questions 5–6
Suppose an array of 2 32-bit ints is written to address 0x007. Assume that we are running on (little-endian) x86 machine and array[0] = `0xba5eba11` and array[1] = `0x5ca1ab1e`.

Question 5 [2 pt]: (see above) What byte is stored at address `0x003`? Write your answer as a hexadecimal number. If the value is outside the array write `unknown`.

Answer: unknown

Question 6 [2 pt]: (see above) What byte is stored at address `0x007`? Write your answer as a hexadecimal number. If the value is outside the array write `unknown`.

Answer: 0x11

Information for questions 7–7
Using notation like $10K$, $2M$, etc. where K, M, etc. represent powers of two.

Question 7 [2 pt]: (see above) Write 2^{23} compactly.

Answer: 8M

Question 8 [2 pt]: Consider the following C function code where `x` is a 32-bit signed integer:

```
if (x < 0) {
    x = x*5;
}
```

Which of the following are equivalent? Assume all right shifts are arithmetic.

A `x |= (x << 2);`
B `x += (x << 2) & (x >> 31);`
C `x += (x << 4) & (x >> 31);`
D `x += (x << 4) & (x >> 30)`
E `x |= (x << 2) & (x >> 30);`
F `x |= (x << 2) ^ (x >> 31);`
G none of the above

Answer: B

Question 9 [2 pt]: Consider the C code below. Which of the following is a correct assembly translation of the function above? (A, B, C, or D)

```
int search(int c){
    while(c < 11){
        c += c;
    }
    return c;
}
```

A.

```
search:
    mov     %edi, %eax
.L3:
    cmp     $10, %eax
    jg      .L1
    add     %eax, %eax
    jmp     .L3
.L1:
    ret
```

B.

```
search:
    mov     %edi, %eax
.L3:
    cmp     $11, %eax
    jle     .L1
    add     %eax, %eax
    jmp     .L3
.L1:
    ret
```

Answer: A

C.

```
search:
    mov     %edi, %eax
    cmp     $11, %edi
    jle     .L2
    xor     %eax, %eax
.L2:
    ret
```

D. none of these

Question 10 [2 pt]: Which of the following are more likely characteristics of RISCS ISAs than CISC ISAs? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] RICS ISAs result in shorter assembly programs

B [ ] RICS ISAas have variable length instructions

C [✓] RICS ISAs more registers

D [ ] Allowing one of the operands in a subtract instruction to be a memory location

Question 11 [2 pt]: Consider the following C code:

```
p = &p[2 * c];
```

where

- `c` is declared as a `long` stored in `%rcx`;
- `p` is declared as a `short *` (pointer to short) stored in `%rax`; and

This is equivalent to which x86-64 assembly instruction?

A `lea (%rax, %rcx, 4), %rax`
B `lea (%rcx, %rax, 2), %rax`
C `mov (%rax, %rcx, 8), %rax`
D `lea (%rax, %rcx, 8), %rax`
E none of the above

Answer: A

Question 12 [2 pt]: Given a 32-bit unsigned integer `x` which of the following C snippets copies the least significant 12 bits of the integer into the second least significant 12 bits? For example, the integer (specified in hexadecimal) `0x12345678` should become `0x12678678`. **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [✓] `x = ((x & ~0xFFFFFF) | (((x & 0xFFF) << 12) | (x & 0xFFF));`
B [ ] `x = ((x & ~0xFFFFFF) | ((x << 12) | x) & 0xFFFFFF);`
C [✓] `x = ((x << 12) & 0xFFF000) + (x & 0xFF000FFF);`
D [ ] `x = ((x ^ (x >> 12)) | ((x << 12) & 0xFFF000));`

Question 13 [2 pt]: In the single-cycle processor design discussed in lecture and our textbook, which of the following instructions use the ALU result? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] jmp
B [✓] call
C [✓] ret
D [ ] nop

Question 14 [2 pt]: Which of the files in the compilation pipeline include the relocation table? (Assume that static linking is used.) **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

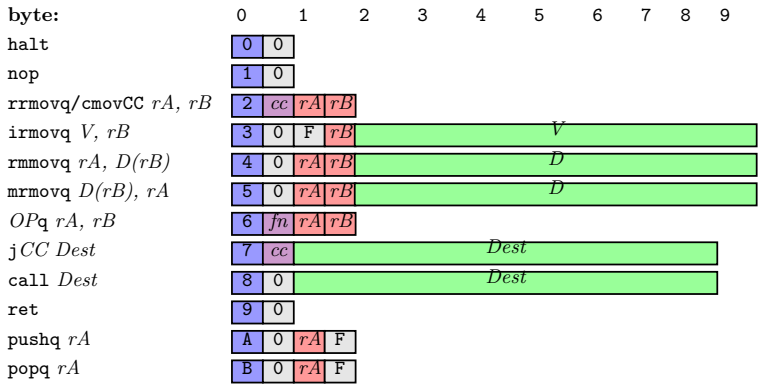
A [✓] .o files

B [ ] .exe files

C [ ] .c files

D [ ] .s files

Information for questions 15–16



Selected register numbers: `%rax`: 0, `%rcx`: 1, `%rdx`: 2 `%rbx`: 3.
Selected OPq `fn` values: `add`: 0, `sub`: 1, `and`: 2, `xor`: 3.

Consider the execution of a Y86 machine where the initial contents of memory are shown below.

On each line, an address is written before the colon; after the colon is a sequence of byte values written in hexadecimal. The first (leftmost) value is located at the address indicated, the second at that address plus 1, and so on.

Any memory location not specified is initially zero.

Execution starts at address `0x00` and continues until a `halt` instruction is reached

Assume all registers are initially zero and write your answers as hexadecimal number.

```
0x00: 50 00 0A 00 00 00 00 00
0x08: 00 00 60 00 70 03 00 00
0x10: 00 00 00 00 00 00 00 00
```

We also accepted 0x4, 0x17
Account for if PC get incremented
After the PC write

Question 15 [2 pt]: (see above) What is the final value of the PC? if some note about being unsure whether the cc field represents the jump is taken (we forgot to supply the table of cc values), accept 0x16

Answer: 0x03

Question 16 [2 pt]: (see above) What is the two least significant bytes of the register `%rax`?

Also Accepted 0xC0

Answer: 0x00C0

Question 17 [2 pt]: Consider the following snippet of HCLRS code below. Assume DO, D1 and S are **1-bit** input signals, and I1, I2, and Out are **1-bit** signals.

```
I1 = D0 & !S;
I2 = D1 & S;
Out = I1 | I2;
```

The code above computes the same value for `Out` as:

A `Out = [ S == 1 : DO; S == 0 : D1; 1 : 0; ];`
B `Out = D0 + D1;`
C `Out = D0 ^ D1;`
D `Out = [ S == 0 : DO; S == 1 : D1; 1 : 0; ];`
E none of the above

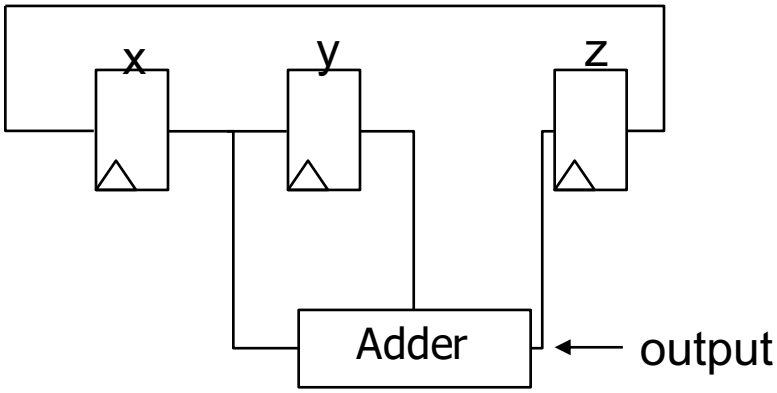
Answer: D

Question 18 [2 pt]: Y86 does not support the x86 instruction `pushq (%rbx)`. Which of the following assembly snippets are equivalent to this instruction? (`;` separates assembly instructions in the answers below.) Assume that `%rax` is a temporary register that can be modified.

A `rmmovq %rsp, 0(%rbx); rrmovq %rsp, %rbx;`
B `mrmovq (%rbx), %rax; pushq %rax`
C `rrmovq %rbx, %rax; add %rsp, %rbx;`
D `pushq %rbx; mrmovq (%rsp), %rbx`
E `rmmovq %rbx, 8(%rsp)`
F none of the above.

Answer: B

Question 19 [2 pt]:



Answer: 2

Consider the above circuit where the box labelled "add" is a combinatorial circuit that performs a 64-bit integer addition, and each of the registers store a 64-bit value and are rising-edge triggered like those we discussed in lecture. If the registers X, Y, Z initially store the values 1, 0, 0 respectively, what is the value of register X after five rising edges of the clock? Write your answer as a base-10 number.

Question 20 [2 pt]: Which of the following are advantages of pipelining? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] less circuitry

B [✓] more instructions completed per unit of time

C [ ] less clock cycles per instruction

D [✓] shorter clock cycles

Question 21 [2 pt]: Suppose one has a single-cycle processor with which takes 2000 ps per cycle. Suppose this processor is evenly divided into four pipeline stages, using registers with 100 ps of register delay for the added registers to support this pipelining. What would the best possible cycle time of the resulting processor be? (Write your answer as a number of picoseconds.)

Answer: 600 ps (also accept 575 ps (rmv PC register delay))

Question 22 [2 pt]: What would we need to change to allow the y86 processor to be able to support the complex address mode used for the second parameter of: `rmmovq %rbx, 8(%rbx, 4)`. **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] the number of inputs to the data memory

B [✓] control logic

C [✓] the ISA

D [ ] the number of inputs and/or outputs to the register file

Pledge:

On my honor as a student, I have neither given nor received aid on this exam.

_____

Your signature here