CS 3330 Exam 1 Fall 2017

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.

. . . .

Question 1 [2 pt]: Each of the following statements about Y86 is true. Which of them is typical of instruction sets that are called RISC-like? **Write "✓" in the box next to each property that is typical of a RISC-like instruction set. Leave other boxes blank.**

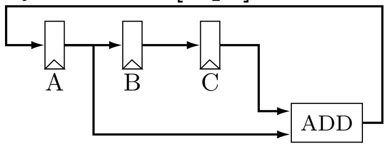
A [✓] the `mrmovq` only has one way to specify a memory location

B [ ] a single instruction can change the stack pointer and read or write from the stack

C [ ] instructions have different lengths, depending on how many operands they have

D [✓] the `addq` instruction does not support a memory operand

Question 2 [2 pt]:



Consider the circuit above, where A, B, and C are 64-bit registers, and ADD is a block of combinatorial logic which takes two 64-bit inputs and produces the 64-bit sum of its inputs. Each of the registers is connected to an (undrawn) common clock signal. If A initially outputs 1, B initilly outputs 2, and C initially outputs 3, what is the value output by A after 2 rising edges of the clock signal? **Write your answer as a decimal number.**

Answer: 6

Question 3 [2 pt]: Consider the following x86 assembly code: `shrq` is a logical right shift, and `andq` is bitwise and.

```
foo:
    andq $0xFF00, %rdi
    shrq $4, %rdi
    movq %rdi, %rax
    retq
```

If this function is executed and `%rdi` (first argument) is `0x12345`, what the value of `%rax` when it returns? Write your answer as a **hexadecimal number**.

Answer: 0x230

Information for questions 4–6
In the single-cycle processor discussed in the book and lecture, the register file has two write register number inputs and two corresponding register value inputs; and two read register number inputs and two corresponding register value outputs. The questions below ask about **pairs** of register file inputs; *each possible answer specifies two values, one for each of the two inputs.*

Question 4 [2 pt]: (see above) While the processor is executing a `call` instruction, what should the two **write register number inputs** be equal to?

A two parts of the instruction memory output
B `0xF` (“no register”) (for both inputs)
C part of the program counter output and the register number for `%rsp`
D ✓the register number for `%rsp` and `0xF` (“no register”) should have been register number for %rsp...
E part of the ALU output and the register number for `%rsp`
F part of the instruction memory output and the register number for `%rsp`

Answer: D

Question 5 [2 pt]: (see above) While the processor is executing a `pushq` instruction, what should the two **read register number inputs** be equal to?

A two parts of the instruction memory output
B part of the ALU output and the register number for `%rsp`
C part of the program counter output and the register number for `%rsp`
D ✓part of the instruction memory output and the register number for `%rsp`
E it doesn’t matter; this instruction doesn’t read registers
F the register number for `%rsp` for one input; and the other doesn’t matter

Answer: D

Question 6 [2 pt]: (see above) While the processor is executing a `subq` instruction, what should the two **read register number inputs** be equal to?

A the register number for `%rsp` for one input; and the other doesn’t matter
should have been register number for %rsp...
B it doesn’t matter; this instruction doesn’t read registers
C part of the ALU output and the register number for `%rsp`
D part of the instruction memory output and the register number for `%rsp`
E ✓two parts of the instruction memory output
F part of the program counter output and the register number for `%rsp`

Answer: E

Question 7 [2 pt]: Suppose we wanted to add a `cmpje rA, rB, DEST` instruction to the Y86 single cycle processor design discussed in class and the textbook. This instruction would test if the value stored in the register `rA` was equal to the value stored in the register `rB`. If so, it would jump to `DEST`. Which of the following would be true about adding this instruction to the processor design? **Write "✓" in each box which is true. Leave other boxes blank.**

A [✓] it would require an extra input to the MUX controlling the program counter input dropped — textbook calls valC whatever the immediate field is, but this comes from a different part of the instruction for instructions with this layout than call/jmp

B [ ] it would be the only instruction that performs an ALU operation without setting the condition codes

C [ ] it would require adding an additional read port to the register file

D [ ] it would require reading more than 10 bytes from the instruction memory at a time

Question 8 [2 pt]: Consider the incomplete Y86 assembly function `absolute_value` below which takes a single 64-bit integer argument in `%rdi` and returns a value in `%rax`:

```
absolute_value:
    irmovq $0, %rax
    subq %rdi, %rax
    ______ %rdi, %rax
    ret
```

In order for this function to return the absolute value of its argument, what should be placed in the blank?

A `subq`
B `cmovg`
C `rrmovq`
D `cmove`
E ✓`cmovl`
F none of the above

Answer: E

Question 9 [2 pt]: What are the values of the condition codes ZF and SF after running the following Y86 assembly snippet:

```
    subq %rax, %rax
    irmovq $-10, %rax
```

A ZF = 0, SF = 0
B ZF = 0, SF = 1
C ZF = 1, SF = 1
D ✓ZF = 1, SF = 0
E not enough information is provided

Answer: D

Question 10 [2 pt]: Using the abbreviations K, M, G, T, etc. (representing powers of two, not ten), what is 2^{17}?

Answer: 128K or M/8

Question 11 [2 pt]: Which of the following are present in object files? **Select all that apply.**

A [ ] the memory addresses of labels

B [✓] the initial values of global variables

C [✓] the names of labels

D [✓] machine code

Question 12 [2 pt]: Consider the following C code:

```
while (b > c) {
    b += c;
}
```

If `b` is stored in `%rbx` and `c` is stored in `%rcx`, which of the following is a correct translation of this loop to assembly code? (Recall that `cmpq A, B` performs the computation `B - A`.) **Write "✓" in the box next to each correct translation. Leave other boxes blank.**

A [ ]

```
    jmp middle
start:
    addq %rbx, %rcx
middle:
    cmpq %rcx, %rbx
    jl start
end:
```

modifies c

B [✓]

```
    cmpq %rcx, %rbx
    jle end
start:
    addq %rcx, %rbx
    cmpq %rbx, %rcx
    jl start
end:
```

C [ ]

```
start:
    addq %rcx, %rbx
    cmpq %rcx, %rbx
    jg start
```

do-while loop

D [✓]

```
start:
    cmpq %rcx, %rbx
    jle end
    addq %rcx, %rbx
    jmp start
end:
```

Information for questions 13–14

Consider the following Y86-64 assembly and its corresponding encoding. The numbers before the `:`s represents a memory address and the sequence of hexadecimal values after it represent the bytes of the instructions in order from lowest to highest address. For example "`0x234: 56 78`" would mean that address `0x234` contains the byte `0x56` and address `0x235` contains the byte `0x78`.

```
0x000: 30 f1 01 00 00 00 00 00 00 00 | irmovq $1, %rcx
0x00a: 30 f2 02 00 00 00 00 00 00 00 | irmovq $2, %rdx
0x014: 62 12                         | andq %rcx, %rdx
0x016: 50 01 08 00 00 00 00 00 00 00 | mrmovq 8(%rcx), %rax
```

Question 13 [2 pt]: (see above) What is the value of `%rax` after this snippet executes?

A `0`
B `1`
C `0x30`
D `0x2f230`
E ✓`0x2f23000`
F `0x30f202`
G `0x30f202000000` (ends with 6 zeroes)
H there is not enough information to determine this
I none of the above

Answer: E

Question 14 [2 pt]: (see above) What is the value of `%rdx` after this snippet executes?

A ✓`0`
B `1`
C `2`
D `3`
E `4`
F there is not enough information to determine this.
G none of the above

Answer: A

Question 15 [2 pt]: Suppose a `long *x` is stored in the register `%rax` and a `long y` is stored in the register `%rbx`. Which of the following x86 assembly snippets is equivalent to the C code `x += y`? (`lea` stands for load effective address.)

A `addq %rbx, %rax`
B `addq (%rbx, 8), %rax`
C ✓`leaq (%rax, %rbx, 8), %rax`
D `movq (%rax, %rbx, 8), %rax`
E `leaq (%rbx, 8), %rax`
F none of the above

Answer: C

Question 16 [2 pt]: The processor design in our book can send part of the output of the instruction memory directly to one of the ALU inputs. Which of these Y86 instructions is this useful for? **Place "✓" in each box corresponding to a true answer. Leave other boxes blank.**

A [✓] `mrmovq`

B [ ] `addq`

C [ ] `call`

D [ ] `pushq`

Information for questions 17–18
Suppose we wanted to add an instruction `iaddq` to the Y86 instruciton set. This instruction would add a constant to an register. For example, `iaddq $100, %rax` would add 100 to the value stored in `%rax`.

Question 17 [2 pt]: (see above) This instruction could have an encoding in machine code with the same layout as:

A ✓`irmovq`
B `mrmovq`
C `addq`
D `call`
E none of the above

Answer: A

Question 18 [2 pt]: (see above) When this instruction executes in the single-cycle design discussed in lecture and our textbook, one of the "value to write" inputs to the register file will not be used, and the other should be equal to:

A the register file output
B the address of the next instruction
C ✓the ALU output
D the data memory output
E none of the above

Answer: C

Question 19 [2 pt]: Consider the following Y86 assembly code:

```
start:
    addq %rax, %rax
    rrmovq %rax, %r8
    subq %rbx, %r8
    jle start
```

If `%rax` corresponds to the variable `a` and `%rbx` corresponds to the variable `b`, which of the following C snippets is equivalent to this code? (You may assume integer overflow does not occur and that all variables are 8-byte signed long ints.)

A `do { a += a; } while (b <= a);`
B `do { a += a; } while (b > a);`
C `do { a += a; a -= b; } while (b <= a);`
D `do { a += a; a -= b; } while (b <= 0);`
E ✓none of the above do-while $a \leq b$

Answer: E

Question 20 [6 pt]: If `x` and `y` are nonnegative unsigned integers between 0 and 10 000 000 (inclusive, decimal numbers), which of the following C expressions are **always** true? **Write "✓" in the box next to each expression which is always true. Leave other boxes blank.**

A		`((x >> 1)	(y >> 1)	1) > (x + y + 1)`
B		`((x & 0xFF) >> 1) > (x & 0xFF)`
C	✓	`(x ^ y) <= (x | y)`
D	✓	`((x >> 7)	(y >> 7)) == ((x	y) >> 7)`
E		`((x & 0xFF)	(x >> 8)) <= 0xFF` dropped for ambiguity re: whether range of numbers was in binary
F		`(x & y) <= (x ^ y)`

Information for questions 21–22
Consider the following C code:

```
int array[5] = {2, 4, 6, 8, 10};
int *x = array + 3;
```

Question 21 [2 pt]: (see above) Immediately after the above code runs, what is the value of `*(x + 1)`?

A ✓10
B 9
C 8
D unknown; this is expression is undefined behavior and/or may crash
E 4
F 6
G ✓none of the above accepted because of variants with confusing question order

Answer: A G

Question 22 [2 pt]: (see above) After `x -= 1; *x -= 1; x -= 1;`, what would the value of `array` be?

A unknown; the code is undefined behavior and/or may crash
B `{2, 3, 6, 8, 10}`
C `{2, 4, 6, 8, 10}`
D ✓`{2, 4, 5, 8, 10}`
E `{2, 4, 6, 7, 10}`
F `{2, 4, 5, 7, 10}`
G none of the above

Answer: D

Pledge:

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

Your signature here