CS 3330 Exam 1 Spring 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.

. . . . . .

Question 1 [3.0 pt]: Suppose an `unsigned char *instr` points to the first byte of a Y86 conditional jmp instruction. In a Y86 conditional jmp instruction, the type of the condition is encoded as a 4-bit number in the least significant bits of the first byte of the instruction. Which of the following C expressions extracts this number? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] `*instr >> 4`

B [ ] `(instr[1] << 4) >> 4`

C [✓] `*instr & 0xF`

D [ ] `(*instr >> 4) << 4`

E [ ] `*instr & 0xFF`

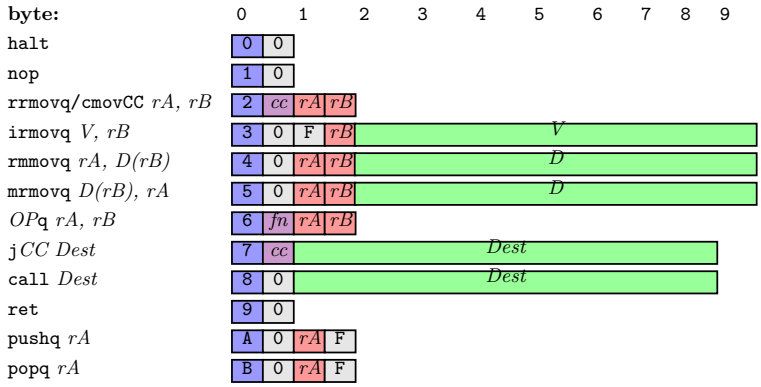
F [ ] `*(instr + 4) & 0xF`

Question 2 [2 pt]: If `a` and `b` are pointers to 32-bit ints which are represented using registers `%rax` and `%rbx` respectively, then which of the following are valid x86-64 assembly snippets are equivalent to the C snippet `b[1+a[0]] = a[0]`? (Semicolons below seperate instructions.)

A `leaq (%rax), %rcx; movl %ecx, 4(%rbx,%rcx,4)`
B `leaq (%rax), %rcx; movl %ecx, 1(%rbx,%rcx,4)`
C `movl (%rax), %ecx; movl %ecx, 1(%rbx,%rcx,4)`
D `leaq 1(%rax,%rbx,4), %rcx; leaq 1(%rcx), %ecx`
E `movl (%rax), %ecx; movl %ecx, 4(%rbx,%rcx,4)`
F `movl (%rax), %ecx; movl %ecx, 4(%rbx,%rax,4)`
G `leaq 1(%rax,%rbx,4), %rcx; movl %eax, (%rcx)`
H none of the above

Answer: E

Question 3 [2 pt]:



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

Answer: 0x246

Referring to the Y86 encoding table and information above, what is the value of `%rax` after the following Y86 machine code (written as a sequence of bytes in hexadecimal) runs? Assume all registers are initially zero and write your answer as a hexadecimal number.

```
30 f0 23 01 00 00 00 00 00 00 60 00 00
```

Question 4 [2 pt]: Using notation like $10K$, $2M$, etc. where K, M, etc. represent powers of two, write 2^{27} compactly.

Answer: 128M

Question 5 [2 pt]: Suppose we wanted to extend the Y86-64 instruction set design to support an `mraddq D(rA), rB` instruction which would add a value from memory to a value in a register and store the result in that register. Which of the following would be true about trying to add support for this instruction in the single-cycle processor design (and keeping it a single-cycle design) described in lecture and our textbook?

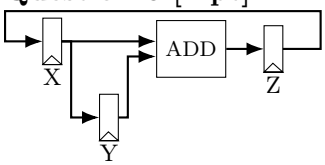
A [ ] it would require an instruction memory that supports reading more bytes at a time

B [ ] it would require an additional input to one of the MUXes that controls what 4-bit register number is sent to the register file's read ports

C [✓] it would require adding an additional ALU or adder

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

Question 6 [2 pt]:



Answer: 4

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, and Z initially store the values 1, 2, and 3 respectively, what is the value of register X after three rising edges of the clock? Write your answer as a base-10 number.

Question 7 [2 pt]: Consider the following C file `foo.c`:

```
int global_variable;
int *foo() {
    return &global_variable;
}
```

From this file, an assembly file `foo.s` and an object file `foo.o` are produced. Then a statically linked executable is created using `foo.o` and `main.o`, an object file which contains a call to `foo()`. The executable's machine code for `foo` contains the memory address of `global_variable` within a `mov` instruction. The **linker** probably produced this by

A computing the address based on the symbol table and/or relocation table of `main.o` and a decision the linker makes about where the contents of `main.o` will be placed in the executable

B copying machine code containing the address within the mov instruction directly from `foo.o`

C computing the address by reading `foo.s` and counting the number of bytes in instructions before the declaration of `global_variable`

D computing the address based on the symbol table and/or relocation table in `foo.o` and a decision the linker makes about where the contents of `foo.o` will be placed in the executable

E copying the address directly from the relocation table or symbol table in `foo.o`

Answer: D

Information for questions 8–9

Suppose a pipelined processor has three stages, which take require the following amounts of time to perform their computations and to store values in registers (including any pipeline registers), memories, etc.:

- Fetch and Decode: 100 picoseconds
- Execute: 75 picoseconds
- Memory and Writeback: 90 picoseconds

Assume this processor always starts a new instruction every cycle and once an instruction is started it always runs to completion.

Question 8 [2 pt]: (see above) What is the minimum cycle time this pipelined processor could have? Write your answer as a base-10 number of picoseconds.

Answer: 100 ps

Question 9 [2 pt]: (see above) If this processor executes a program with 10 instructions, how long will it take from the time the first instruction starts until when the clock cycle for the last instruction's memory and writeback stage finishes? Write your answer as a base-10 number of picoseconds.

Answer: 1200 ps

Information for questions 10–11
Suppose we wanted to extend the Y86-64 instruction set design to support an `iaddq $CONSTANT, rB` instruction which would add a constant (that would be part of the machine code for the instruction) to a register's value and store the result in that register.

Question 10 [2 pt]: (see above) Which of the following would be true about adding this instruction in the single-cycle design (and keeping it a single-cycle design) described in lecture and our textbook? Assume the addition of the constant and register value will be performed using the existing ALU. **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☐ it would require adding a MUX or an additional input to an existing MUX that controls what 64-bit value is input to one of the register file's write ports

B ☐ it would require adding an additional read port in the register file

C ☐ it would require adding a connection from the data memory's output to the ALU

D ☐ it would require adding an additional type of operation to the ALU

Question 11 [2 pt]: (see above) This new instruction could have the same layout in machine code (placement of fields like register numbers and immediates) as

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

Answer: A

Question 12 [2 pt]: Consider the following C function that attempts remove every other element of a singly linked list:

```
struct node { int value; struct node *next; };
void remove_every_other(struct node *head) {
    struct node *p = head; // (1)
    while (p != NULL && p->next != NULL) {
        struct node *t;
        t = p->next;
        p->next = p->next->next;
        p = p->next;
        free(t);
    }
}
```

Which of the following are true about this code? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] `t` must be `malloc()`ed before `p->next` is assigned to it, or the function could segfault

B [ ] `t->value` should also be free'd, or the function will leak memory

C [ ] an assignment to `p->value` is needed after the assignment to `p->next` to `p` for the function to behave correctly

D [✓] if the line marked `(1)` were removed, and the parameter `head` was renamed to `p`, then the function would behave the same way

Question 13 [3.0 pt]: Which of the following are generally attributes of an *instruction set architecture*? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] the number of read ports present in the register file that holds registers accessible to assembly

B [✓] which instruction will be executed when several zero bytes are encountered in machine code

C [✓] whether `nop` and `add` instructions are the same length in machine code

D [✓] the size of registers accessible to assembly was originally miskeyed (and apparently more ambiguous than anticipated); credit given for answers either way *outside of TPEGS* on 1 March (see gradebook)

E [ ] the number of clock cycles required for an `add` instruction

F [✓] whether an `add` instruction can set an "overflow flag" condition code

Question 14 [2 pt]: In the single-cycle processor design discussed in our textbook and in lecture, to read 8 bytes as part of an instruction like `mrmovq`, the processor

A requires the data memory to support reading 8 bytes all at once even though each of those bytes has their own address
B uses word addresses, such that two consecutive 8 byte values have addresses that differ by 1
C uses the ALU to send the addresses of each of the 8 bytes separately to the data memory
D uses the data memory to read some of the bytes, and the instruction memory to read the rest

Answer: A

Question 15 [2 pt]: Consider the following x86-64 assembly code:

```
     add %rcx, %rbx
 top:
     jl bottom
     add $1, %rcx
     sub $1, %rbx
     jmp top
 bottom:
```

Recall that in AT&T syntax assembly, `sub A, B` computes `B - A` and stores the result in `B`. If `%rcx` is represented by the variable `c` and `%rbx` by the variable `b`, this code is equivalent to which of the following C snippets?

A `b += c; while (b >= 0) { c += 1; b -= 1; }`
B `b += c; while (b > 0) { c += 1; b -= 1; }`
C
```
b += c;
if (b < c) { do { c += 1; b -= 1; } while (b >= 0); }
```
D `b += c; do { c += 1; b -= 1 } while (b < 0);`
E `b += c; do { c += 1; b -= 1 } while (b >= 0);`
F none of the above

Answer: A

Question 16 [2 pt]: In the single-cycle processor design discussed in our textbook and in lecture, when executing an `mrmovq D(rA), rB` instruction, the PC register will be updated to contain the address of the following instruction _____.

A at around the same time as when `D(rA)` is computed
B at around the same time the `mrmovq` instruction is read from memory
C potentially any time after `rB` is updated and before the next instrution is executed
D at around the same time `rA` is read from the register file
E at around the same time as when `rB` is updated in the register file

Answer: E

Question 17 [2 pt]: In the single-cycle processor design discussed in our textbook and in lecture, when executing an `rmmovq rA, D(rB)` instruction, the ALU output

A has no value
B is used to form an input to the data memory
C has a value, but is ignored because MUXes will select other values
D is used to determine a 4-bit destination register number for the register file
E is used to form an input to the register file

Answer: B

Question 18 [2 pt]: Given signed ints `x` and `y` between `-9999` and `9999` where negative numbers are represented using two's complement, which of the following C expressions should always be true? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] `((x & 0xFF) * 4) == ((x << 2) & 0xFF)`

B [ ] `~((~x) | (~y)) == (x|y)`

C [ ] `((~x) & y) < ((~x) | y)`

D [✓] `((x & 0xF0) >> 4) == (((x & 0xF7) >> 4) & 0x7F)`

Question 19 [2 pt]: In the single-cycle processor design discussed in our textbook and in lecture, when executing an `rmmovq rA, D(rB)` instruction, the two 4-bit register number to read inputs of the register files will be equal to

A part of the instruction memory's output and part of the ALU's output
B part of the instruction memory's output and the register number of %rsp
C part of the data memory's output and `0xF` (the "no register" value)
D part of the instruction memory's output and part of the data memory's output
E `0xF` and `0xF` (the "no register" value)
F two different parts of the instruction memory's output
G part of the instruction memory's output and `0xF` (the "no register" value)

Answer: F

Question 20 [2 pt]: Which of the following are likely attributes of an instruction set that follows the RISC (reduced instruction set computer) design philosophy? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] using a variable-length encoding that makes simpler instructions have a shorter encoding

B [✓] only allowing register operands in `add` instructions to simplify the hardware

C [ ] allowing all instructions to access memory, avoiding mov-like instructions to access memory

D [ ] including a dedicated string copying instruction

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

```
short a[6] = {1,2,3,4,5,6};
short *p;
p = &(a[1]);
*p += a[2];
p += a[2];
*p = 0;
```

Question 21 [2 pt]: (see above) The value of `a` after this code runs is

A `{1,2,3,4,0,6}`
B `{1,0,3,4,5,6}`
C `{1,5,3,4,0,6}`
D `{1,5,3,4,2,6}`
E none of the above; the code stays in bounds of the array `a`, but produces a different result
F none of the above; the code goes out of bounds of the array `a` or dereferences a NULL pointer

Answer: C

Question 22 [2 pt]: (see above) If `a` is located starting at memory address `0x1000` on a little-endian system with 2-byte shorts then, the value of the byte of memory at `0x1005` before any assignments to `p` execute is _____. (Write your answer as a hexadecimal number. If there is not enough information, write "unknown")

Answer: 0

Pledge:

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

_____

Your signature here