CS 3330 Exam 1 Spring 2018

Name: _____ Computing ID: _____

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]: Suppose one added a new instruction `immovq $V, D(rB)` to the Y86-64 ISA and the single-cycle implementation we discussed in lecture. This instruction would move a 64-bit constant from the instruction into memory at a particular location, where the memory location was specified exactly the same way as for `rmmovq`. Which of the following is true about implementing this instruction in the single-cycle processor we discussed in lecture? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☐ either `V` or `D` would need to appear in a different part of the machine code than for other Y86-64 instructions

B ☐ it would require an output larger than 10 bytes from the instruction memory

C ☐ it would require a data memory with the ability to read and write in the same cycle

D ☐ it would require an additional input to the register file

Information for questions 2–3

Consider the following Y86 program and its corresponding encoding. (On the leftmost column is the memory address of each instruction, followed by its encoding, listed as a sequence of bytes in hexadecimal, with the leftmost byte being the one at the lowest address.)

```
0x000: 63 00                            | xorq %rax, %rax
0x002: 50 30 0b 00 00 00 00 00 00 00 | mrmovq 0xB(%rax), %rbx
0x00c: 62 30                            | andq %rbx, %rax
0x00e: 00                               | halt
```

Assume all unlisted bytes of memory contain `0`.

Question 2 [2 pt]: (see above) What is the value of `%rbx` when this snippet halts?

A `0x62 3000 0000 0000` (ends with 11 zeroes)
B `0x62`
C `0xb30500063`
D `0xb30`
E `0x300b 0000 0000 0000` (ends with 12 zeroes)
F `0`
G `0x306200`
H `0x6300 5030 0b00 0000`
I `0x30620000`
J `0x6230`
K none of the above

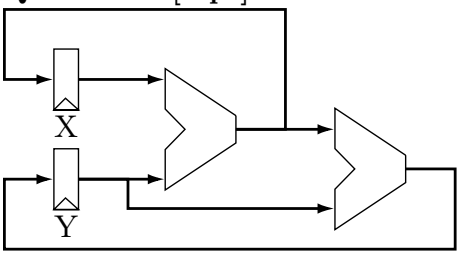
Answer:

Question 3 [2 pt]: (see above) What is the value of the condition codes `SF` and `ZF` when this snippet halts?

A `SF = 1` and `ZF = 1`
B `SF = 1` and `ZF = 0`
C `SF = 0` and `ZF = 0`
D `SF = 0` and `ZF = 1`
E not enough information is provided

Answer:

Question 4 [2 pt]:



Answer:

Consider the above circuit containing two 64-bit registers `X` and `Y` and ALUs that only perform 64-bit additions. If the registers both initially output `1`, what value will be output of the register `Y` after two rising edges of the clock signal? Write your answer as a base-10 number.

Question 5 [2 pt]: Consider the following C function that allocates and returns a singly-linked list:

```
struct node { int value; struct node *next; };
struct node *makeList() {
    struct node *p = malloc(sizeof(struct node) * 2);
    p[0].value = 1;
    p[0].next = &p[1];
    p[1].value = 2;
    p[1].next = NULL;
    return p;
}
```

Allocating a linked list this way is problematic. Which of the following statements about this code are true? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☐ Passing a pointer to second node of the returned list to `free()` is illegal or invokes undefined behavior.

B ☐ It is possible for the caller of `makeList` to free the `malloc`ed memory.

C ☐ Passing a pointer to first node of the returned list to `free()` is illegal or invokes undefined behavior.

D ☐ The returned list can be converted to a normal array of two `int`s by casting the returned pointer to an `int*`.

Question 6 [2 pt]: If `unsigned char *instr` points to a Y86 instruction `addq rA, rB`, which of the following C expressions results in the register number corresponding to `rA` (i.e. `0` for `%rax`, `1` for `%rcx`, etc.)? Recall that this register number is the most significant half of the second byte of the instruction. **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☐ `instr[1] >> 4`

B ☐ `(instr[0] >> 12) & 0xF`

C ☐ `instr[1] & 0xF0`

D ☐ `(instr[1] & 0x0F) >> 4`

Question 7 [2 pt]: Using notation where K, M, G, etc. represent powers of two, not ten, what value does 64K represent? Write your answer in the form 2^X where X is an base-10 integer.

Answer:

Question 8 [2 pt]: If a `long **x` is assigned to register `%r8`, which of the following assembly snippets is equivalent to the C snippet `x += **x;`?

A
```
movq (%r8), %rax
movq (%rax), %rax
leaq (%r8,%rax,8), %r8
```

B
```
leaq (,%r8,8), %rax
movq (%rax), %rax
addq %rax, %r8
```

C
```
movq (%r8), %rax
movq (%rax), %rax
leaq (,%rax,8), %r8
```

D
```
movq (%r8), %rax
leaq (%r8,%rax,8), %rax
addq %rax, %r8
```

E
```
movq (%r8), %rax
movq (%rax), %rax
addq %rax, %r8
```

Answer:

Question 9 [2 pt]: Suppose the Y86-64 ISA was modified by changing the encoding of the `pushq` instruction from the 2-byte encoding:

- byte 0: 4-bit opcode `0xA` and 4-bit constant `0x0`
- byte 1: 4-bit register number and 4-bit constant `0xF`

to a 1-byte encoding:

- byte 0: 4-bit opcode `0xA` and 4-bit register number

In which of the following ways would this change affect the single-cycle Y86-64 design we discussed in lecture? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☐ It would make it substantially harder to compute instruction lengths.

B ☐ It would require additional MUXes, inputs to MUXes, or other logic between the instruction memory and register file.

C ☐ It would require an additional kind of ALU operation.

D ☐ It would require that a register file that has more inputs.

Question 10 [2 pt]: Which of the following pieces of information must be present in an object file in order for an executable to be produced from it? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☐ the names of functions that whose machine code is in other object files

B ☐ the memory addresses of functions whose machine code is in other object files

C ☐ the type (`int`, `char` array, etc.) of each constant in the object file's data section

D ☐ the locations within the object file that correspond to labels used in other object files

Question 11 [2 pt]: When the single-cycle processor design described in lecture is executing the `jmp` instruction, the register file's "64-bit value to write" inputs will

A be equal to the next value for the PC (the target of the `jmp`)
B not have any value
C be equal to the value of the PC
D be ignored because there will be no rising edge in the clock signal
E be ignored because corresponding register number inputs will have the value 15 ("no register")

Answer:

Question 12 [2 pt]: Suppose one was designing an ISA for a system with very limited storage space for machine code. Which of the following ISA design decisions would be likely to reduce the storage required for machine code? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☐ including instructions that perform arithmetic or logic directly on values in memory

B ☐ using variable-length instruction encodings

C ☐ including instructions that correspond directly to high-level operations, like sorting

D ☐ providing several thousand general purpose registers instead of several tens of registers

Question 13 [2 pt]: Consider the following assembly code:

```
    testq %r12, %r12
    jmp middle
start:
    movq %r12, %rdi
    call foo
    subq %rax, %r12
middle:
    jge start
```

Assuming the variable `long x` is mapped to the callee-saved register `%r12`, which of the following C snippets could this have been produced from? (Recall that `%rdi` holds the first argument to a function and `%rax` holds a function's return value.)

A `do { x -= foo(x); } while (x >= 0);`
B `while (x >= foo(x)) {}`
C `while (x >= 0) { x -= foo(x); }`
D `do { x -= foo(x); } while (x < 0);`
E `while (x < 0) { x -= foo(x); }`
F none of the above

Answer:

Question 14 [2 pt]: In the single-cycle processor design described in lecture, the amount of time between when we start executing instruction A and when we start executing the next instruction B: **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☐ varies depending on the length of the clock cycle

B ☐ varies depending on how much computation instruction A needs to perform

C ☐ varies depending on whether instruction B reads registers written by instruction A

D ☐ varies depending on whether instruction A is a conditional jump instruction or not

Question 15 [2 pt]: If `x`, `y`, and `z` are `unsigned int`s, Which of the following C expressions are **always true**? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☐ `((x + y) >> 7) == (x >> 7) + (y >> 7)`

B ☐ `(x & y & z) <= (x | y | z)`

C ☐ `((x >> 4) & 0xFF) == ((x & 0xFF8) >> 4)`

D ☐ `(((x >> 2) | x) & y) == ((x & y) | ((x & y) >> 2))`

Question 16 [2 pt]: When the single-cycle processor design we described in lecture executes an `rrmovq` instruction, one of the two "4-bit number of register to write" inputs to the register file is:

A equal to part of the output of the ALU
B equal to `4` (the register number for %rsp)
C equal to part of the output of the data memory
D equal to part of the output of the register file
E equal to part of the output of the instruction memory

Answer:

Question 17 [2 pt]: In the single-cycle processor design described in lecture, when the instruction `popq %rbx` is executed, the value stored in the register file for `%rbx` is updated _____ the value of `%rsp`.

A at about the same time as
B after
C unknown; it depends on how fast the data memory is
D before
E none of the above

Answer:

Question 18 [2 pt]: Which of the following are true about RISC-like processor designs compared to CISC-like processor designs? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☐ RISC-like designs give hardware designers more flexibility to optimize performance

B ☐ RISC-like designs usually implement more instructions than CISC-like designs

C ☐ RISC-like designs make it easier to write a correct compiler for a high-level language

D ☐ RISC-like designs allow new hardware versions to be developed more quickly

Information for questions 19–21
Consider the following C code:

```
short array[5] = {1,2,3,4,5};
short *p;
p = array + 2;
*p = 4;
p += 1;
*p += p[1];
```

Question 19 [2 pt]: (see above) If `short`s are 2 bytes and all pointers are 8 bytes, what is the value of `sizeof(array)`?

A 2
B 8
C 10
D 12
E 18
F 20
G none of the above

Answer:

Question 20 [2 pt]: (see above) On a little endian system where `short`s are 2 bytes, if the array starts at address `0x10000`, what is the value of the **byte** at address `0x10002` just after the array is initialized?

A `0`
B `1`
C `2`
D `3`
E `4`
F none of the above

Answer:

Question 21 [2 pt]: (see above) What is the final value of `array`?

A the code invokes undefined behavior and may crash
B `{1, 4, 7, 4, 5}`
C `{1, 2, 4, 5, 6}`
D `{1, 2, 3, 4, 5}`
E `{1, 2, 4, 9, 5}`
F none of the above

Answer:

Pledge:

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

_____

Your signature here