CS 3330 Exam 1 – Fall 2015

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

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: Labels are part of assembly, but not part of the underlying ISA the assembly represents. This is because

A Labels are turned into addresses by the assembler or linker
B Labels are used by a different part of the hardware, not the ISA
C Labels are like comments; they have no semantic meaning
D None of the above

Answer:

Information for questions 2–4
Consider a six-bit IEEE-style floating-point number with 1 sign, 2 exponent, and 3 fraction bits. Assume the bias is 1. Answers are written in binary.

Question 2: (see above) What is the largest denormalized value?

A 1110
B 11110
C 111
D 0.0111
E 0.01111
F 0.111
G 1111
H 0.1111

Answer:

Question 3: (see above) How many of the 2^6 bit patters in this format are NaNs?

A 2, 3, or 4
B 0
C 1
D 5, 6, or 7
E 8
F between 9 and 15
G 32
H between 17 and 31
I 16

Answer:

Question 4: (see above) What is the smallest non-negative normalized value?

A 1
B 0.0001
C 0.1
D 0
E 100
F 10
G 0.001

Answer:

Information for questions 5–7

Consider adding a new `mmmovq` instruction to Y86-64 that moves from memory to memory. Assume we use the same operand notation for `mmmovq` that we use for other operations.

Question 5: (see above) How many program registers will `mmmovq` need to access?

A 1
B 0
C 2
D more than 2

Answer:

Question 6: (see above) The data memory functionality we use for simulating Y86-64 has one output (`rvalM`, the value retrieved from memory when reading) and four inputs:

- `dread`, a bit meaning "I want to read memory"
- `dwrite`, a bit meaning "I want to write memory"
- `addr`, 64 bits meaning "the address of memory to access"
- `wvalM`, 64 bits meaning "the value to put into memory when writing"

Which of the following is true of the requirements for adding `mmmovq`?

A We'd need one or more new inputs and one or more new outputs
B The existing inputs and outputs would be sufficient
C We'd need one or more new outputs, but the existing inputs would be sufficient
D We'd need one or more new inputs, but the existing outputs would be sufficient

Answer:

Question 7: (see above) How many bytes are needed to encode `mmmovq`? To avoid double-coverage of the register counting question in this set, assume that it takes R bytes to encode the register(s) needed (e.g., for `nop` $R = 0$, for `irmovq` and `OPq` $R = 1$, etc).

A $10 + R$
B $1 + R$
C $9 + R$
D $2 + R$
E none of the above

Answer:

Information for questions 8–11
Consider the C statement `*x = y + **z;`. Assume that the compiler chooses to store `x`, `y`, and `z` each in a different program register, and that `y` and `z` are already in their registers.

Question 8: (see above) How many Y86-64 `rmmovq`s are needed for that statement?

A 0
B 3
C 4
D 2
E 1

Answer:

Question 9: (see above) How many Y86-64 `OPq`s are needed for that statement?

A 0
B 2
C 3
D 4
E 1

Answer:

Question 10: (see above) How many Y86-64 `mrmovq`s are needed for that statement?

A 0
B 3
C 4
D 1
E 2

Answer:

Question 11: (see above) How many Y86-64 `rrmovq`s are needed for that statement?

A 0
B 4
C 2
D 3
E 1

Answer:

Information for questions 12–17

Assume `x` and `y` are non-negative `int` values. For answers where one or both is the operand of a shift, assume it has a legal value for shifting with.

For each question, write a relational operator (<, <=, ==, !=, >=, >) in the box that will make the expression true for all values of `x` and `y`. Note that if < is true, so is <= and `!=`; in that case, write <. If none of the relational operators is true for all `x` and `y`, write `none`.

For example, if the question was `x` ____ `x | ~x` I'd answer > because `x` is non-negative and `x | ~x` is always −1.

Question 12: (see above) `(x < y) ? ((x << y)` ____ `(y << x)) : ((y << x)` ____ `(x << y))` (assume the same relational operator has to go in both blanks)

Answer:

Question 13: (see above) `(x & y)` ____ `(x | y)`

Answer:

Question 14: (see above) `(x + y)` ____ `(x << y)`

Answer:

Question 15: (see above) `(x + y)` ____ `((x ^ y) + ((x & y) << 1))`

Answer:

Question 16: (see above) `(x ^ y)` ____ `(x + y)`

Answer:

Question 17: (see above) `x` ____ `(x + y)`

Answer:

Question 18: Suppose you have a system that does three tasks sequentially: task A takes 60% of the time, task B 30%, and task C 10%. Which of the following would give the largest speedup?

A Running A, B, and C all in parallel without changing their respective runtimes but with a 2% cost splitting and recombining the tasks
B Cutting all three tasks to $\frac{2}{3}$ of their current times
C Cutting B's time to $\frac{1}{10}$ of its current time
D Cutting A's time to $\frac{1}{3}$ of its current time

Answer:

Information for questions 19–22
Consider the following stage summaries:

- Fetch interacts with instruction memory and computes the new PC
- Decode reads values from the register file
- Execute does any math not done in Fetch
- Memory interacts with data memory
- Writeback writes values to the register file

and the following subset of the Y86-64 operations: `irmovq`, `rmmovq`, `mrmovq`, `OPq`, `jXX`, `call`, `ret`, `pushq`, and `popq`. (This list does *not* include `halt`, `nop`, `rrmovq`, or `cmovXX`).

Question 19: (see above) How many operations from that list do need the memory stage but not the execute stage?

A 2 or 3
B 1
C 0
D more than 3

Answer:

Question 20: (see above) How many operations from that list do need the memory stage but not the writeback stage?

A 0
B more than 3
C 1
D 2 or 3

Answer:

Question 21: (see above) Which of the following instructions (a) uses both execute and memory and (b) would work if the memory stage ran before the execute stage?

- **A** `rmmovq`
- **B** `mrmovq`
- **C** `popq`
- **D** `pushq`
- **E** two or more of the above
- **F** none of the above

Answer:

Question 22: (see above) How many operations from that list do not need the decode stage?

- **A** 0
- **B** more than 3
- **C** 1
- **D** 2 or 3

Answer:

Question 23: The following Y86-64 code leaves what value in `%rax`?

```
irmovq $0x1234, %rbx
rmmovq %rbx, 0x100
irmovq $0x5678, %rbx
rmmovq %rbx, 0x108
mrmovq 0x104, %rax
```

Answers are shown in hex in 2-byte clusters for readability

- **A** 0x0000 0000 0000 0000
- **B** 0x0000 0000 5678 0000
- **C** 0x0000 7856 0000 0000
- **D** 0x0000 0000 3412 0000
- **E** 0x0000 1234 0000 0000
- **F** None of the above

Answer:

Information for questions 24–28

In lab 2 we worked with three kinds of lists; repeating those definitions, suppose we have the following defined:

```
typedef struct node_t { TYPE value; node *next; } node;
typedef struct range_t { size_t length; TYPE *ptr; } range;
TYPE *sentinel_array;
node *linked_list;
range length_array;
```

Question 24: (see above) On a 64-bit machine where `sizeof(TYPE)` is 4 and `sizeof(size_t)` is 8, which requires the least memory for a 50-element list?

A `sentinel_array`
B `linked_list`
C `length_array`

Answer:

Question 25: (see above) Assume `TYPE` is `int`. I have a list [2,1,5,0, 3,3,3,0, 4,4,1,4] and want to "split on 0" to get the three lists [2,1,5], [3,3,3], and [4,4,1,4]. If I *require the original list to remain untouched*, which list type(s) can do that without any additional `malloc` calls?

Select all that apply by putting one or more letter in the box

A `sentinel_array`
B `length_array`
C `linked_list`
D none of the above

Answer:

Question 26: (see above) Assume `TYPE` is `int`. I have a list [2,1,5,0, 3,3,3,0, 4,4,1,4] and want to "split into chunks of 4 elements" to get three lists [2,1,5,0], [3,3,3,0], and [4,4,1,4]. If I *am OK with losing the original list*, which list type(s) can do that without any additional `malloc` calls?

Select all that apply by putting one or more letter in the box

A `linked_list`
B `length_array`
C `sentinel_array`
D none of the above

Answer:

Question 27: (see above) Assume `TYPE` is `int`. I have a list [2,1,5,0, 3,3,3,0, 4,4,1,4] and want to "split on 0" to get the three lists [2,1,5], [3,3,3], and [4,4,1,4]. If I *am OK with losing the original list*, which list type(s) can do that without any additional `malloc` calls?

Select all that apply by putting one or more letter in the box.

A `sentinel_array`
B `linked_list`
C `length_array`
D none of the above

Answer:

Question 28: (see above) In C, strings are implemented as a list of `char`s for which type of list?

A `sentinel_array`
B `linked_list`
C `length_array`

Answer:

Question 29: If `x` is a binary integer then ∼`x + 1` is the same as `-x`; this encoding is called "two's compliment." There is another encoding where ∼`x = -x`, called "one's compliment." One's compliment has two zeros (like floating-point does); assuming we still do binary addition normally, it also has which of the following other problems?

(Note: there is a simple tweak to addition that fixes this problem, but not one we will explore)

A `6 + -3` is not three
B `3 + -3` is not zero
C `3 + -5` is not negative two
D `0 + -0` is not zero

Answer:

Question 30: What is the minimal number of `jXX` operations needed to assemble `if(a) b = c;` into Y86-64?

A 2
B 0
C 1
D 3

Answer:

Pledge:

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

____

Your signature here