CS 3330 Exam 1 – Spring 2016

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.
- $\sim$ means bitwise-negation, $\wedge$ means exclusive-or
- questions are single-selection

Multiple-select: are all clearly marked; put 1 or more letters in the box, or write "None" if no options are correct.
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: Given a 6-bit IEEE-style floating-point number with two fraction bits, which of the following is the largest denormalized value we can represent? Answers are shown in base 2

A 1.1
B 0.000011
C 0.00011
D 11
E 0.011
F $+\infty$
G 0.11
H 0.0011

Answer: H

Information for questions 2–3
Given the following definitions from lab 2

```
typedef struct range_t { size_t length; double *ptr; } range;
typedef struct node_t { double value; node *next; } node;
```

and the terminology:

- a linked list is stored as a `node *`
- a sentinel array is stored as a `double *` and uses NaN as the sentinel
- a length array is stored as a `range`

Question 2: (see above) Which list type conversions listed below can be performed without modifying heap memory or allocating new heap memory?
Select all that apply

A linked list → length array
B linked list → sentinel array
C sentinel array → length array
D sentinel array → linked list
E length array → sentinel array
F length array → linked list

Answer: C

Question 3: (see above) In a list containing four values, which of the following uses the least *heap*-allocated space?

A the length-carrying array `range`
B the sentinel-terminated array `double *`
C the linked-list `node *`

Answer: A

Question 4: Given an 8-bit IEEE-style floating-point number with at least one fraction bit, what is the minimum number exponent bits needed in order to have the largest finite float be larger than the largest `signed char`?

A 4 max 11110000
B 1 max 0.111111
C 2 max 11.1111
D 3 max 1111.1
E 5
F 6
G None of the above are sufficient

Answer: A

Question 5: Suppose that instead of condition codes we added a register operand to conditional operations (`jXX` and `cmovXX`), where the new operand is compared to 0 to see if the conditional operation should proceed. Thus we'd write asm like "`jg %rax, L2`" with semantics like the C code "`if (rax > 0) goto L2;`".

The current Y86-64 has four kinds of information layouts in their encodings: 1-, 2-, 9-, and 10-byte versions. This change in the assembly would

A add a fifth layout jXX now 10, cmovXX new 3-byte
B add a new layout but also remove one, leaving us at 4
C not change the number of layouts, leaving us at 4
D add more than one new layout
E remove the need for one of the layouts, reducing us to 3

Answer: A

Question 6: Task X takes 2 microseconds; task Y takes 3 microseconds; and task Z takes 5 microseconds. Running them as-is sequentially takes $X + Y + Z = 10$ microseconds. Which of the following would result in a speed-up of at least $\frac{5}{3}\times$ compared to that baseline?
Select all that apply

A Do the three sequentially but speed up Z by $4\times$
B do X and Y in parallel; then do Z, speeding up Z by $2\times$
C Do an extra 1.5 microsecond task; then do X, Y, and Z all in parallel
D X and Y sequentially, with Z in parallel with them

Answer: B D

Question 7: In x86-64 and Y86-64, when you execute a `push` the value in the stack pointer

A becomes larger
B stays the same
C becomes smaller
D it depends on the value being pushed

Answer: C

Information for questions 8–10
Suppose an array of two `shorts` is written to address `0x204`. Assume `array[0] = 0x1234` and `array[1] = 0x5678`. Assume the rest of memory contains 0 bytes.

Question 8: (see above) What is the value of the byte stored in address `0x204`?

A `0x87`
B `0x56`
C `0x43`
D `0x34`
E `0x78`
F `0x12`

Answer: D

Question 9: (see above) What is the value of the byte stored in address `0x201`?

A `0x00`
B `0x34`
C `0x56`
D `0x12`
E `0x65`
F `0x78`
G `0x21`

Answer: A

Question 10: (see above) What is the value of the byte stored in address `0x207`?

A `0x78`
B `0x65`
C `0x34`
D `0x56`
E `0x21`
F `0x00`
G `0x12`

Answer: D

Question 11: `call` is sometimes described as a `pushq` that pushes the PC instead of an operand register followed by a `jmp`. Why have a special opcode for that; why not simply write `pushq %pc; jmp address`?

A `call` and `pushq` modify the stack pointer by different amounts
B `call` has effects not present in `pushq`+`jmp` that are not listed above
C `call` pushes a offset of the PC, not the PC directly
D `pushq`+`jmp` has effects not present in `call` that are not listed above
E none of the above

Answer: C

Question 12: An unconditional C `goto address` statement is compiled as an assembly `jmp address` and in Y-86 binary as a nine-byte sequence `70` *addressIn8Bytes*. A C label like `L2:` is

A is a label in assembly and is a label in binary
B is a label in assembly and is in binary but not as a label
C is in assembly but not as a label and is in binary but not as a label
D is a label in assembly and is not in binary
E is not in assembly and is in binary
F is in assembly but not as a label and is not in binary

Answer: D

Question 13: Write $log_2(128\text{M})$ as base-ten number

Answer: 27

Question 14: A 32-bit signed integer can store ____ numeric values compared to a 32-bit unsigned integer

A fewer
B more
C the same number of

Answer: C

Question 15: Given a 6-bit IEEE-style floating-point number with two fraction bits, which of the following is the smallest non-negative normalized value we can represent? Answers are shown in base 2

A 0.0001
B 0.1
C 0.00001
D 0.001
E 0.01

Answer: E

Question 16: `ret` is sometimes described as a `popq` that writes the popped value into the PC instead of into the operand register. Why have a special opcode for that; why not simply write `popq %pc`?

A `ret` offsets the PC in addition to popping it
B `ret` and `popq` modify the stack pointer by different amounts
C `ret` has additional effects not present in `popq` that are not listed above
D `popq` has additional effects not present in `ret` that are not listed above
E none of the above

Answer: E

Question 17: A 32-bit float can store ____ numeric values compared to a 32-bit signed integer

A more
B the same number of
C fewer

Answer: C

Information for questions 18–22
For each of the following, assume that `x` and `y` are declared in C as `int`s; that `f` and `g` are declared as `float`s; and that `u` and `v` are declared as `unsigned int`s. Assume that all size values are initialized to positive (non-zero, non-negative, finite) values.

Each of the following present two expressions with a blank between them and gives a list of comparators. **Select all** options for which the resulting expression could be `true` for some value of the variables. For example, `x __ 1`'s correct answer is > and == but not <.

Question 18: (see above) `~!~x __ -1`
(those ~ are large tildes, meaning bitwise-not in C; **Select all that apply**)

A <
B ==
C >

Answer: B

Question 19: (see above) `f + g __ f`; assume no overflow occurs
(**Select all that apply**)

A >
B == underflow
C <

Answer: A B

Question 20: (see above) `x + y __ 0`
(**Select all that apply**)

A ==
B <
C >

Answer: B C

Question 21: (see above) `u ^ v __ u + v`; assume no overflow occurs
(that ^ is a large carat, meaning xor in C; **Select all that apply**)

A >
B ==
C <

Answer: B C

Question 22: (see above) `u + v __ 0`
(**Select all that apply**)

A >
B ==
C <

Answer: A B

Question 23: Y86-64's memory operands' most complicated form is `immediate(%register)`, but X86-64 also has `immediate(%register1,%register2,8)`. The last number (`8` in the example) can only be `1`, `2`, `4`, or `8` which means it can fit in 2 bits (4 options fits in 2 bits).

If we added the more complicated memory operand syntax to `mrmovq`, how many bytes long would the new instruction need to be? (Do not modify `mrmovq`'s `icode` or `ifun`).

A 10
B 11
C 1
D 12
E 9
F 2
G 13

Answer: B

Question 24: Y86-64 assembly has two-operand `OPq`s (`subq rA, rB`), but C also has three-operand versions (`c = b - a`). To get the three-operand semantics (having the result and both operands available after the operation) in Y86-64 assembly requires

A three assembly instructions
B three assembly instructions and an extra register for a temporary value
C one assembly instruction
D two assembly instructions
E one assembly instruction and an extra register for a temporary value
F two assembly instructions and an extra register for a temporary value

Answer: D

Pledge:

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

Your signature here