CS 3330 Final Exam – Fall 2016

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.
- `sub A, B` performs `B -= A`, not `A -= B`
- questions are single-selection unless identified as select-all. If none of the answers of a select-all are correct, either leave the box blank or write "none" in the box.

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.

. . . .

ⓐ ⓑ ⓒ ⓓ ⓔ ⓕ ⓖ ⓗ ⓘ ⓙ ⓚ ⓛ ⓜ ⓝ ⓞ ⓟ ⓠ ⓡ ⓢ ⓣ ⓤ ⓥ ⓦ ⓧ ⓨ ⓩ
ⓐ ⓑ ⓒ ⓓ ⓔ ⓕ ⓖ ⓗ ⓘ ⓙ ⓚ ⓛ ⓜ ⓝ ⓞ ⓟ ⓠ ⓡ ⓢ ⓣ ⓤ ⓥ ⓦ ⓧ ⓨ ⓩ
ⓐ ⓑ ⓒ ⓓ ⓔ ⓕ ⓖ ⓗ ⓘ ⓙ ⓚ ⓛ ⓜ ⓝ ⓞ ⓟ ⓠ ⓡ ⓢ ⓣ ⓤ ⓥ ⓦ ⓧ ⓨ ⓩ
⓪ ① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨
ⓐ ⓑ ⓒ ⓓ ⓔ ⓕ ⓖ ⓗ ⓘ ⓙ ⓚ ⓛ ⓜ ⓝ ⓞ ⓟ ⓠ ⓡ ⓢ ⓣ ⓤ ⓥ ⓦ ⓧ ⓨ ⓩ
ⓐ ⓑ ⓒ ⓓ ⓔ ⓕ ⓖ ⓗ ⓘ ⓙ ⓚ ⓛ ⓜ ⓝ ⓞ ⓟ ⓠ ⓡ ⓢ ⓣ ⓤ ⓥ ⓦ ⓧ ⓨ ⓩ

Do not write in this area

Department		Course			Exam	Page	Version
○	○	●	○	○	●	●	●
●	○	●	●	○	○	○	○
○	○	●	○	○	○	○	○
●	○	●	○	○	●	○	○
○	○	○	○	○	○	○	○
●	○	●	○	○	○	○	○

Information for questions 1–4
Suppose a system uses 8192-byte (2^{13} bytes) pages and a two-level page table, where each level of the page table has one page of 4-byte page table entries with the following format:

bit 0	present (not on disk)
bit 1	readable/executable
bit 2	writable
bits 6–29	physical page number
other bits	not relevant to these questions

(Bit 0 is the least significant bit of the page table entry when it is interpreted as an integer.)

Question 1 [2 pt]: (see above) What is the size in bits of a physical address on this platform?

Answer:

Question 2 [2 pt]: (see above) If the last-level page table entry for the virtual address `0x12345678` has the value `0x103` and the processor is trying to read an instruction from that address, what memory address will be read?

A `0x1678`
B `0x3678`
C `0x4678`
D `0x9678`
E none; the access will result in a page fault
F none of the above

Answer:

Question 3 [2 pt]: (see above) What is the size in bits of a virtual address on this platform?

Answer:

Question 4 [2 pt]: (see above) If this system has a 8-entry, 4-way set associative TLB, then accessing the address `0x12345678` requires accessing the same set of the TLB as:

A `0x12399789`
B `0x1234FFFF`
C `0x8678`
D `0x7F665F02`
E `0xFF6789`
F none of the above

Answer:

Question 5 [2 pt]: In each box, put `Y` if the code will always print `yes` or `N` if it won't. You may assume the code is syntactically valid and does not experience errors.

A ☐ `if(0.0 == 0) printf("yes");`

B ☐ `char[] s = {'y', 'e', 's'}; printf(s);`

C ☐ `printf("yes" ? "yes" : "no");`

D ☐ `if(3330) printf("yes");`

Question 6 [2 pt]: In the the following Y86-64 program:

```
    irmovq $1, %rax
    irmovq $16, %rbx
loop:
    addq %rax, %rax
    rrmovq %rax, %rcx
    subq %rbx, %rcx
    jne loop
```

How many times does the `addq` instruction execute?

A 2
B 8
C 16
D 4
E 1
F none of the above

Answer:

Question 7 [2 pt]: If we change the number of lines per set in a cache, without changing the number of sets, number of bytes per block, or bits in the address, which of the following change length? **Select all that apply.**

A tag
B index
C offset

Answer:

Question 8 [2 pt]: Which of the following cause the processor to enter kernel mode? **Select all that apply.**

A traps
B signals
C faults
D interrupts
E some non-privileged code that is not included in any of the above
F some priviledged code that is not incldued in any of the above

Answer:

Question 9 [2 pt]: Consider a system with 4096-byte pages and 4-level page tables. If a virtual address `0x7FFF0000` maps to a physical address `0x9000` for a process on this system, then which are of the following are always true? **Select all that apply.**

A virtual address `0x7FFEFFFF` maps ot physical address `0x8FFF`
B the process has at least 16384 bytes (4 times 4096) of page tables
C the process has at least 524288 bytes (128 times 4096) of page tables.
D virtual address `0x7FFF0001` maps to physical address `0x9001`

Answer:

Question 10 [2 pt]: Which of the following are reasons why optimizing compilers may not automatically unroll loops? **Select all that apply.**

A unrolling loops requires identifying pointer aliasing
B unrolling loops can increase the number of instruction cache misses
C unrolling loops can require spilling registers to the stack
D unrolling loops is only beneficial for large loops
E unrolling loops can increase the number of data cache misses

Answer:

Information for questions 11–12
Suppose we have an implementation of Y86-64 where

- The execute stage is slowest when adding, requiring 1 ns to complete
- The memory stage is slowest overall, requiring 1.5 ns to complete
- All other stages are faster than those two

Assume we add a new `ifun` to `OPq`: `mulq`, which requires 1.8 ns in the execute stage.

Question 11 [2 pt]: (see above) If this is a five-stage pipelined processor, how much slower will our clock cycle become? Assume we don't change the stages other than adding this new operation to the ALU.

A it won't need to change
B it will need to be 0.3 ns slower
C it will need to be 0.8 ns slower
D it will need to be 1.8 ns slower

Answer:

Question 12 [2 pt]: (see above) If this is a sequential processor, how much slower will our clock cycle become?

A it won't need to change
B it will need to be 0.3 ns slower
C it will need to be 0.8 ns slower
D it will need to be 1.8 ns slower

Answer:

Information for questions 13–15
Consider the following Y86-64 code:

```
irmovq $0x10, %rcx
irmovq $0x10000, %rax
addq %rcx, %rcx
mrmovq 0(%rax), %rbx
subq %rax, %rcx
```

Suppose a **page fault** occurs while the `mrmovq` instruction is executed.

Question 13 [2 pt]: (see above) Suppose the page fault handler loads the page that caused the page fault from disk and then updates the corresponding page table entries. What instruction should be run when the page fault handler returns?

A `irmovq $0x10, %rcx`
B `irmovq $0x1000, %rax`
C `addq %rcx, %rcx`
D `mrmovq 0(%rax), %rbx`
E `subq %rax, %rcx`

Answer:

Question 14 [2 pt]: (see above) Before the page fault handler is run, what will the value of the condition codes `ZF` (zero flag) and `SF` (sign flag) be?

A ZF = 0 and SF = 0
B ZF = 0 and SF = 1
C ZF = 1 and SF = 1
D ZF = 1 and SF = 0

Answer:

Question 15 [2 pt]: (see above) If the system uses 4096-byte pages and a 4-level page table with 4-byte page table entries and the page table base register is set to **page number** `0x10`, what was the **address** of the *first-level* page table entry accessed before the page fault?

A `0x10`
B `0x11`
C `0x12`
D `0x10004`
E `0x10008`
F `0x11000`
G not enough information to answer
H none of the above

Answer:

Information for questions 16–17
Consider the following Y86-64 assembly snippet:

```
irmovq $0x5000, %rax
irmovq $0x2000, %rbx
irmovq $0x4000, %rcx
mrmovq 5(%rax), %rax
addq %rax, %rbx
subq %rcx, %rbx
```

Question 16 [2 pt]: (see above) In order to avoid more pipeline stalls, our pipelined processor needs to forward between which of the following pairs of instructions? **Select all that apply.**

A `irmovq $0x5000, %rax` and `addq %rax, %rbx`
B `irmovq $0x5000, %rax` and `mrmovq 5(%rax), %rax`
C `mrmovq 5(%rax), %rax` and `addq %rax, %rbx`
D `mrmovq 5(%rax), %rax` and `subq %rcx, %rbx`
E `addq %rax, %rbx` and `subq %rcx, %rbx`

Answer:

Question 17 [2 pt]: (see above) With the pipelined processor implementation we constructed in homeworks, if `irmovq $0x5000, %rax` is fetched in cycle 1, in what cycle will `subq %rcx, %rbx` write its result to a register?

Answer:

Question 18 [2 pt]: Which of the following statements are **always true** if `x` and `y` are integers between 0 and 4096 inclusive? Mark `Y` if the expression is always true and `N` if it is not.

A		`x & y <= x ^ y`
B		`-x > ~x`
C		`(x >> 2) < x`
D		`x + x == x << 1`

Question 19 [2 pt]: Which of the following snippets of C code has semantics most similar to the following Y86-64 assembly code?

```
rrmovq %rdi, %rax
subq   %rsi, %rdi
cmovg  %rsi, %rax
```

```
A  rax = (rdi < rsi ? rsi       : rdi      );
B  rax = (rdi < rsi ? rdi       : rdi - rsi);
C  rax = (rdi < rsi ? rdi - rsi : rdi      );
D  rax = (rdi < rsi ? rdi       : rsi      );
```

Answer:

Information for questions 20–22
Some commands in Y86-64 use a specific register, such as `ret` using `%rsp`. Suppose we wanted to make all such commands have an explicit register argument instead (e.g., `ret %rsp`).

As a reminder, there are 12 Y86-64 instructions: `nop`, `halt`, `irmovq`, `cmovXX` (which includes `rrmovq`), `rmmovq`, `mrmovq`, `OPq`, `jXX`, `pushq`, `popq`, `call`, and `ret`.

Question 20 [2 pt]: (see above) How many instructions would need a different length encoding as a result of this change?

Answer:

Question 21 [2 pt]: (see above) How many instructions would this ISA change impact?

Answer:

Question 22 [1 pt]: (see above) Would this change require any new encoding scheme beyond the 1-byte, 2-byte, 9-byte, and 10-byte versions we have already? Answer "yes" or "no"

Answer:

Question 23 [2 pt]: Consider a data cache with 256 128-byte blocks on a system with 4096-byte pages. (4096 is 32 times 128.) In order for the index and block offset bits for this cache to be contained entirely within the page offset bits of an address, what is the *minimum* associativity the cache must have?

Answer:

Question 24 [2.5 pt]: Answer True or False for each of the following.

A		If the ISA had only one kind of **interrupts**, software could emulate the other kinds by setting a register value before triggering the interrupt
B		If the ISA didn't have **traps**, software could emulate them by setting a register value and then triggering a **fault**
C		If the ISA had only one kind of **fault**, software could emulate the other kinds by setting a register value before triggering the fault
D		If the ISA didn't have **traps**, software could emulate them by setting a register value and then triggering an **interrupt**

Question 25 [2 pt]: Suppose a processor has a one multiply functional unit, which has an issue rate of 1 per cycle and a latency of 2 cycles.

How much faster or slower will `((a * b) * c) * d)` be than `((a * b) * (c * d))`?

A the same speed
B 2 cycles faster
C 1 cycle slower
D 2 cycles slower
E 1 cycle faster
F none of the above

Answer:

Information for questions 26–28
Answer the following questions about different kinds of exceptions.

Question 26 [2 pt]: (see above) Which of the following are true about **faults**? **Select all that apply.**

A The handler for a fault must restart the interrupted program.
B The handler for a fault will run in kernel mode.
C The operating system can ensure that a program gets a fault after executing for a certain amount of time.
D All faults are deliberately caused by a program.
E On Linux, a fault can result in a signal handler running.

Answer:

Question 27 [2 pt]: (see above) Which of the following are true about **interrupts**? **Select all that apply.**

A The handler for an interrupt must restart the interrupted program.
B The handler for an interrupt will run in kernel mode.
C The operating system can ensure that a program gets an interrupt after executing for a certain amount of time.
D All interrupts are deliberately cause by a program
E On Linux, an interrupt can result in a signal handler running.

Answer:

Question 28 [2 pt]: (see above) Which of the following are true about **traps**? **Select all that apply.**

A The handler for a trap must restart the interrupted program.
B The handler for a trap will run in kernel mode.
C The operating system can ensure that a program gets a trap after executing for a certain amount of time.
D All traps are deliberately caused by a program.
E On Linux, a trap can result in a signal handler running.

Answer:

Question 29 [3 pt]: Put letters from the options below in the following boxes to finish the following sentences.

To return from a **signal** handler we use

To return from a **hardware exception** handler we use

A you can't return from these
B an ordinary `jmp` instruction
C a trap
D a larger block of non-trap instructions together
E an ordinary `ret` instruction
F an ordinary `call` instruction

Information for questions 30–30
Consider the following two versions of a C function:

```
float sum1(float *array, int len) {
    float result = 0.0f;
    for (int i = 0; i + 1 < len; i += 2) {
        result += array[i];
        result += array[i+1];
    }
    return result;
}

float sum2(float *array, int len) {
    float result1 = 0.0f, result2 = 0.0f;
    for (int i = 0; i + 1 < len; i += 2) {
        result1 += array[i];
        result2 += array[i+1];
    }
    return result1 + result2;
}
```

Assume `len` is larger than 10.

Question 30 [2 pt]: (see above) Which version is likely to be faster?

A `sum1`
B `sum2`
C their performance is likely to be the same

Answer:

Question 31 [2 pt]: Consider a direct-mapped cache with 256 sets and 16 byte blocks. In this cache the address `0x12345` maps to the same set as which of the following addresses? **Select all that apply.**

A `0x12340`
B `0x22244`
C `0x02345`
D `0xF434F`
E `0x12354`

Answer:

Question 32 [2.5 pt]: Put letters from the options below in the following boxes to finish the sentence.

A signal is like ☐ except it is generated by ☐ instead of ☐ and is handled by ☐ instead of ☐.

A the processor
B a fault
C kernel code
D user code
E an interrupt
F a trap

Question 33 [2 pt]: Various caches we discussed are noteworthy for having only one of something. Write in the blank what each has only one of

A Direct-mapped caches have only one ________________ per ________________

B Fully-associative caches have only one ________________ per ________________

Question 34 [3 pt]: Suppose we use a processor counter to see how many assembly instructions are executed. Which of the following optimizations will reduce that number?
Select all that apply.

A reassociation of operators
B loop unrolling
C moving work outside of loops
D loop reordering
E loop blocking
F function inlining

Answer:

Question 35 [2 pt]: Consider a 4-way set associative, write-back, write-allocate, 16KB data cache with 32 byte blocks and an LRU replacement policy. Given an address `x` represented as an unsigned integer, which of the following expressions gives the set index number of the address in the system's cache?

A `(x & 0xFFF) >> 5`
B `(x >> 5) & 0xFE`
C `x & 0xFE0`
D `(x >> 5) & 0xFE0`
E `x & 0xFFF`
F none of the above

Answer:

Question 36 [2 pt]: Suppose we change our 5-stage pipeline into a 10-stage pipeline by cutting each stage's work exactly in half. Assume that the data needed in stage X is now needed in stage X_1 and that values produced or set in stage X are now produced or set in stage X_2.

Which of the following would need stalling in the 10-stage pipeline but didn't need stalling in the 5-stage pipeline?

Select all that apply.

A two `jXX` in a row with different `ifun`s
B an `mrmovq` followed by an `OPq` with the same destination register
C two `OPq`s in a row with the same destination register
D two `popq`s in a row with different destination registers

Answer:

Question 37 [2.5 pt]: Why do caches use the middle bits of the address for the set index instead of the high-order bits? **Select all that apply**.

A Doing so allows the cache to take better advantage of spatial locality.
B Doing so decreases the amount of space required to store cache tags.
C Doing so allows the cache to take better advantage of temporal locality.
D Doing so makes it easier to overlap the TLB lookup with the cache lookup.
E Doing so ensures that adjacent blocks in memory map to different cache sets.

Answer:

Information for questions 38–37

Question 38 [2 pt]: Suppose we ran the following code:

1. `void *ptr = malloc(16);`
2. `long x = (long)ptr;`
3. *this line varies by answer*
4. `void *ptr2 = (void *)x;`

Assuming we have 4KB pages, which of the following options for line 3 would be most likely to have `ptr` and `ptr2` on different pages of memory?

A 3. `x &= 0xff;`
B 3. `x |= 0xff;`
C 3. `x ^= 0xff;`
D 3. `x += 0xff;`

Answer:

Question 39 [2 pt]: Consider the following C code:

```
void multiply(float *x, float *y, float *z, int len) {
    for (int i = 0; i < len; ++i)
        x[i] = (y[i] + x[i]) * z[i];
}
```

In order for it to be correct for a compiler to store the result in `x[i]` after loading values for the computation of `x[i+1]`, which of the following are **sufficient** conditions? **Select all that apply.**

A this is never safe
B `(x != y) && (x != z)`
C this is always safe
D none of the values pointed to by `x`, `y`, or `z` are infinite or NaN
E `x`, `y`, and `z` were allocated with different calls to `malloc`

Answer:

Information for questions 40–41
Consider the following C snippet:

```
unsigned char array[1024 * 1024];
...
for (int iteration = 0; iteration < 2; ++iteration) {
    for (int i = 0; i < 4096; i += 8) {
        array[i] += 1;
    }
}
```

Assume that only accesses to `array` use the data cache (all other values are kept in registers) and that `array[0]` is stored in the first byte of a cache block.

Question 40 [2 pt]: (see above) If this snippet is run with an initially empty 3KB, 3-way set associative data cache with 32-byte blocks and an LRU replacement policy, how many cache misses will occur?

A 640 (512 + 128)
B 0
C 1024
D 512
E 4096
F none of the above

Answer:

Question 41 [2 pt]: (see above) If this snippet is run with an initially empty 2KB, direct-mapped data cache with 32-byte blocks, how many cache misses will occur?

A 1024
B 4096
C 0
D 512
E 640 (512 + 128)
F none of the above

Answer:

Information for questions 42–43
Suppose we add a divide instruction to the 5-stage pipelined Y86-64 processor. The ALU needs 15 cycles to produce the result of a division. While the ALU is computing the result of the division, the pipeline must be stalled.

Question 42 [2 pt]: (see above) What should the value "bubble" and "stall" inputs to the pipeline registers be during this stall? Answer by putting an N (for normal), B (for bubble), or S (for stall) in each pipeline register's box.

	F		D		E		M		W

Question 43 [2 pt]: (see above) Suppose instead of using a pipeline stall, we added fifteen additional execute stages to our pipeline, resulting in 20 stage pipeline. For how many cycles would a `ret` instruction cause a stall on this modified pipeline?

Answer:

Pledge:

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

____
Your signature here