CS 3330 Exam 2 Spring 2017

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 [4 pt]: Suppose we are executing the following assembly code on the 5-stage pipelined processor we discussed in class:

```
        irmovq $0x1, %rax
        andq   %rax, %rax
        jle           later // untaken branch
        irmovq $0x5, %rax
        subq   %rcx, %rax
 later: addq   %rax, %rbx
        halt
```

When should the stall and bubble control signals of our pipeline register banks be set to when the `jle` instruction is in its execute stage in order to cause the `addq` and `halt` instructions to be squashed? (The box before `F` represents the register holding the predicted PC.)

Write `B` to indicate that the bubble signal is set; `S` to indicate that the stall signal is set; and `N` to indicate neither are set.

N	F		D		E		M		W

Information for questions 2–4

Suppose we modified our five-stage pipelined processor into a six-stage pipelined processor by splitting the memory stage into two parts, so the resulting stages are:

- Fetch
- Decode
- Execute
- Memory 1
- Memory 2
- Writeback

Assume that the address at which to access memory and what value to write to the data memory must be available near the beginning of the Memory 1 stage and that any value read from the data memory will not be available until near the end of the Memory 2 stage. Assume that this processor implements **all possible forwarding paths**.

Question 2 [2 pt]: (see above) Suppose this processor executes the following sequence of instructions in this order:

1. `addq %rax, %rbx`
2. `popq %rcx`
3. `subq %rbx, %rcx`

If the `addq` insturction executes its Fetch stage during cycle 0, then during what cycle will the `subq` instruction execute its Writeback stage?

Answer:

Question 3 [2 pt]: (see above) Suppose this processor executes the following sequence of instructions in this order:

1. `pushq %rax`
2. `popq %rax`

If the `pushq` insturction executes its Fetch stage during cycle 0, then during what cycle will the `popq` instruction execute its Writeback stage?

Answer:

Question 4 [2 pt]: (see above) Suppose this processor executes the following sequence of instructions in this order:

1. `addq %rax, %rbx`
2. `subq %rcx, %rdx`
3. `ret`
4. `xorq %r8, %r9`

If the `addq` instruction executes its Fetch stage during cycle 0, then during what cycle will the `xorq` instruction execute its Writeback stage?

Answer:

Question 5 [2 pt]: Suppose you developed the next greatest memory technology, MagicRAM. The access latency of a MagicRAM cell is 2 times that of a SRAM but lower than the access latency a DRAM. MagicRAM has higher density than DRAM.

Assume that you have a system that has a 64KB L1 cache made of SRAM, a 12MB L2 cache made of SRAM, and 4GB main memory made of DRAM. Where would you put your magicRAM in the memory hierarchy to get better performance? **Select all that apply.**

A Replace main memory with magicRAM memory
B Replace L1 SRAM cache with L1 magicRAM cache
C Add another level of L3 cache made of magicRAM cells
D Replace L2 SRAM cache with L2 magicRAM cache

Answer:

Question 6 [2 pt]: Suppose you have narrowed down the design choices for the L1 cache of the next-generation processor to the following two designs:

- Choice 1: A 64KB cache with 64-byte blocks
- Choice 2: A 64KB cache with 8-byte blocks

Both of them have the **same number of sets**. Which statements about these designs are definitely true? **Select all that apply.**

A 1 has more bits of storage for tags
B 1 is better at exploiting temporal locality
C 1 has more blocks
D 1 is better at exploiting spatial locality

Answer:

Question 7 [2 pt]: Suppose, instead of directly using a number of address bits to index into the cache (and choose what set to access), a cache took those bits and used a hash function implemented in hardware to decide which index in the cache to access. What type of cache misses can this idea potentially reduce?

A Capacity misses
B Compulsory misses
C Carbon misses
D Conflict misses

Answer:

Question 8 [2 pt]: Which of the following is likely to occur if we *increase* the size of a cache without increasing its block size or associativity? **Select all that apply.**

A the number of tag bits stored per set will increase
B the hit rate will increase
C the hit latency will increase
D the number of conflict misses will decrease

Answer:

Question 9 [2 pt]: Which of the following statements about *precise exceptions* are true? **Select all that apply.**

A A single-cycle processor has precise exceptions.
B Implementing precise exceptions makes debugging easier.
C To implement precise exceptions, an out-of-order processor must track some instructions after their computation has completed.

Answer:

Question 10 [3 pt]: Consider executing the following sequence of instructions with the five-stage pipeline design we discussed in lecture:

```
pushq %rsp
rmmovq %rsp, 4(%rax)
```

Which of the following are **possible** forwarding paths for the value of `%rsp` between these two instructions? **Select all that apply.**

A from the memory stage of `pushq` to the execute stage of `rmmovq`
B from the decode stage of `pushq` to the execute stage of `rmmovq`
C from the writeback stage of `pushq` to the execute stage of `rmmovq`
D from the execute stage of `pushq` to the decode stage of `rmmovq`
E from the memory stage of `pushq` to the decode stage of `rmmovq`
F from the execute stage of `pushq` to the execute stage of `rmmovq`

Answer:

Question 11 [2 pt]: Consider an out-of-order processor executing the following sequence of instructions:

```
mrmovq 0(%rcx), %rbx
subq %rcx, %rdx
andq %rbx, %r9
xorq %r9, %rcx
addq %rdx, %rax
```

Which **calculations** could an out-of-order processor complete before the `mrmovq` instruction completes? **Select all that apply.**

A `subq %rcx, %rdx`
B `xorq %r9, %rcx`
C `andq %rbx, %r9`
D `addq %rdx, %rax`

Answer:

Question 12 [4 pt]: The pipelined Y86-64 processor we discussed in lecture can execute

```
addq %rax, %rbx
rmmovq %rbx, 4(%rax)
mrmovq 4(%rbx), %rax
nop
nop
subq %rbx, %rax
```

without stalling. Which of the following forwarding needs to occur for this to happen? **Select all that apply.**

A `%rbx` is forwarded from `addq` to `rmmovq`
B `%rbx` is forwarded from `addq` to `mrmovq`
C `%rbx` is forwarded from `addq` to `subq`
D `%rax` is forwarded from `addq` to `rmmovq`
E `%rax` is forwarded from `addq` to `mrmovq`
F `%rbx` is forwarded from `rmmovq` to `mrmovq`
G `%rax` is forwarded from `mrmovq` to `subq`
H `%rbx` is forwarded from `mrmovq` to `subq`

Answer:

Question 13 [2 pt]: A set-associative cache breaks a 32-bit address into a 20-bit tag and a 4-bit block offset. How many sets are there in this cache?

A 256
B 8
C 16
D 3
E none of the above

Answer:

Information for questions 14–16
Consider a processor with

- a 256B, 2-way L1 (level 1) cache with 5-cycle access latency with 64 byte blocks; and
- a 16-way set-associative L2 cache with a 20-cycle access latency that holds 128 64-byte blocks

The replacement policy for both of the caches are true LRU. The average memory access time (AMAT) of a cache is $AMAT = (\text{hit-rate} \times \text{hit-latency}) + (\text{miss-rate} \times \text{miss-latency})$.

Question 14 [2 pt]: (see above) A programmer writes a program that repeatedly accesses (in a loop) only **two** unique cache blocks. The loop is executed for billions of iterations. In the steady state (after many iterations of the loop), what do you expect the average memory access time to be?

A 20 cycles
B 5 cycles
C 10 cycles
D 7.5 cycles

Answer:

Question 15 [2 pt]: (see above) What is the access time when a block misses in L1 cache, but hits in L2 cache?

A 10 cycles
B 20 cycles
C 25 cycles
D 5 cycles

Answer:

Question 16 [2 pt]: (see above) A programmer writes a program that in a loop repeatedly accesses **eight** unique cache blocks in a fixed order. (An access pattern like A, B, C, D, E, F, G, H, A, B, C, D, ...) The loop is executed for billions of iterations. In the steady state (after many iterations of the loop), what do you expect the average memory access time to be?

A 10 cycles
B 7.5 cycles
C 20 cycles
D 25 cycles

Answer:

Information for questions 17–20

Consider the following two versions of C code, where `SIZE` is a large integer.

Version A:
```
for (int j = 0; j < SIZE; ++j)
  for (int i = 0; i < SIZE; ++i)
    array1[i * SIZE + j] += array2[i * SIZE + i] - array3[j * SIZE + i];
```
Version B:
```
for (int i = 0; i < SIZE; ++i)
  for (int j = 0; j < SIZE; ++j)
    array1[i * SIZE + j] += array2[i * SIZE + i] - array3[j * SIZE + i];
```

Question 17 [1 pt]: (see above) Which version has substantially better *spatial* locality in the accesses to `array1`?

A A
B B
C they are about the same

Answer:

Question 18 [1 pt]: (see above) Which version has substantially better *temporal* locality in the accesses to `array1`?

A A
B B
C they are about the same

Answer:

Question 19 [1 pt]: (see above) Which version has substantially better *spatial* locality in the accesses to `array2`?

A A
B B
C they are about the same

Answer:

Question 20 [1 pt]: (see above) Which version has substantially better *temporal* locality in the accesses to `array2`?

A A
B B
C they are about the same

Answer:

Pledge:

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

____

Your signature here