CS 3330 Exam 2 – 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.
- questions are single-selection unless identified as select-all

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: Which of the following caches has the largest blocks? (due to ambiguity, 0 points)

A A full-associative cache with 6-bit tags and 128 lines
B A direct-mapped cache with 8-bit tags and 64 lines
C A direct-mapped cache with 6-bit tags and 128 lines
D A full-associative cache with 8-bit tags and 64 lines
E All of the above have the same block size
F Which of the above depends on the address size

Answer: A

Information for questions 2–4
Given a 12-bit address `jklmnopqrstu`, where `j` is the high-order bit, what are the parts of the address for a **direct-mapped** cache with 32 lines and 8-byte blocks?
Answer by putting one or more letters in the box; e.g., `klmn` would be the second-, third-, fourth-, and fifth-highest-order bits

Question 2: (see above) The index is (1 point; 0.5 for size, 0.5 for place)

Answer: nopqr

Question 3: (see above) The tag is (1 point; 0.5 for size, 0.5 for place)

Answer: jklm

Question 4: (see above) The offset is (1 point; 0.5 for size, 0.5 for place)

Answer: stu

Information for questions 5–9
Suppose we expand our pipeline to ten stages as follows:

- Fetch has three stages; the PC is needed at the beginning of F_1; `valP` (that is, old PC + offset) is available at the end of F_1; and the parts of the instruction (`icode`, `valC`, etc) are available at the end of F_3.
- Decode is still one stage.
- Execute is two stages; operands are needed at the beginning of E_1 and results available at the end of E_2. Condition code checking for conditional jumps and moves is performed in E_2.
- Memory is three stages; the address and whether to read or write is needed at the beginning of M_1; the value to write is needed at the beginning of M_2; and a value read from memory is available at the end of M_3.
- Writeback is still one stage.

In the following questions, **assume we forward** instead of stalling wherever possible. Answer each question with an integer between 0 and 10.

Question 5: (see above) How many stages of speculative work would be erased on a branch misprediction? (2 points, 1 for 3 or 6)

Answer: accept 4 and 5

Question 6: (see above) We said `valP` was available in F_1. Why would a version where `valP` was not available until F_2 be a very bad idea? Write a short answer here:

we'd have to stall every cycle; we'd have to speculate every instruction; etc. (2 points)

Question 7: (see above) How many cycles of stalling would be needed to resolve a (memory) load-use hazard? (2 points, 1 for 3 or 5)

Answer: 4

Question 8: (see above) How many cycles of stalling would be needed to run

```
addq %rax, %rbx
addq %rbx, %rcx
```
(2 points)

Answer: 1

Question 9: (see above) How many cycles of stalling would a `ret` hazard create? (2 points, 1 for 7 or 9)

Answer: 8

Question 10: Of all of the Y86-64 instructions, only `popq` uses both write ports `E` and `M` in the register file, one for `%rsp` and one for storing the value popped from the stack. Each of the other instructions use at most one register-file write port.

`ret` is similar to `popq` in many ways; why does `ret` not need two register file write ports? (2 points)

A It does not update the stack pointer
B It updates the stack pointer, but not a register in the register file
C The value it pops from the stack is stored, but not in the register file
D The value it pops from the stack is not stored

Answer: C

Information for questions 11–13
The following questions are grouped because they have the same set of possible answers.

Question 11: (see above) Which of the following optimizations can change the big-O of the code? (2 points; 1 if they redefine big-O to defend a bad answer)

A loop unrolling
B function inlining
C multiple accumulators
D moving work outside of loops
E none of the above

Answer: D

Question 12: (see above) Which of the following optimizations is designed to take advantage of pipelines? (2 points)

A loop unrolling
B moving work outside of loops
C multiple accumulators
D function inlining
E none of the above

Answer: C

Question 13: (see above) Which of the following optimizations is designed to take advantage of caches? (2 points)

A function inlining
B loop unrolling
C multiple accumulators
D moving work outside of loops
E none of the above

Answer: E

Information for questions 14–15
In the following diagrams, indicate the control signals to give each pipeline register by putting a single letter in each box; use N for normal, B for bubble, and S for stall.

Question 14: (see above) Assume that i_4 and i_5 resulted from incorrect speculative execution and should not be allowed to continue but all other instructions are OK and may continue to execute normally. (2 points, -0.5 per wrong letter)

Stage:		F		D		E		M		W
Contains:		i_5		i_4		i_3		i_2		i_1
Answers:	N		B		B		N		N	

Question 15: (see above) Assume that i_3 requires some information not yet available and needs another cycle in the E stage, but all other instructions are OK and may continue to execute normally. (2 points, -0.5 per wrong letter)

Stage:		F		D		E		M		W
Contains:		i_5		i_4		i_3		i_2		i_1
Answers:	S		S		S		B		N	

Question 16: When running Y86-64 instructions on the single-stage SEQ implementation described in the textbook and implemented in the first HCL homework, we may encounter (2 points)

A dependencies
B hazards
C both of the above
D neither of the above

Answer: A

Question 17: Longer pipelines tend to achieve what compared to shorter pipelines? **Select all that apply** (2 points, 0.5 per option)

A lower throughput
B higher latency
C lower latency
D higher throughput

Answer: B D

Question 18: Assume we have a 4-line direct-mapped cache. If we access the following addresses (split into parts for you) in the order listed, which will hit and which will miss? Answer by putting an H or M in each box.(3 points, 0.5 per box)

	Tag	Index	Offset
M	0	0	0 accept H if mention cache could be warm
H	0	0	8
M	1	0	0
M	0	3	8 accept H if mention cache could be warm
M	0	0	0
H	0	3	0

Information for questions 19–21
Given a 12-bit address `jklmnopqrstu`, where `j` is the high-order bit, what are the parts of the address for a **set-associative** cache with 8 sets of 4 lines each and 8-byte blocks?

Answer by putting one or more letters in the box; e.g., `klmn` would be the second-, third-, fourth-, and fifth-highest-order bits

Question 19: (see above) The index is (1 point; 0.5 for size, 0.5 for place)

Answer: pqr

Question 20: (see above) The tag is (1 point; 0.5 for size, 0.5 for place)

Answer: jklmno

Question 21: (see above) The offset is (1 point; 0.5 for size, 0.5 for place)

Answer: stu

Information for questions 22–23
The following questions ask about how specific characteristics of caches.

Question 22: (see above) Suppose I know that memory accesses in my program will be purely sequential, with addresses being accessed in order.
Which of the following will have the largest impact on memory performance? (2 points)

- **A** high levels of associativity
- **B** total cache size
- **C** number of individual blocks
- **D** individual block size

Answer: D

Question 23: (see above) Suppose I know that memory accesses in my program will be completely random, with no temporal or spatial locality.
Which of the following will have the largest impact on memory performance? (2 points)

- **A** total cache size
- **B** number of individual blocks
- **C** high levels of associativity
- **D** individual block size

Answer: A

Question 24: Which of the following code snippets exhibits the most spatial locality? (2 points)

A
```
for(c=0; c<n; c+=1)
  for(r=0; r<n; r+=1)
    a[r][c] += b[r][c];
```
B
```
for(r=0; r<n; r+=1)
  for(c=0; c<n; c+=1)
    a[r][c] += b[c][r];
```
C
```
for(c=0; c<n; c+=1)
  for(r=0; r<n; r+=1)
    a[r][c] += b[c][r];
```
D
```
for(r=0; r<n; r+=1)
  for(c=0; c<n; c+=1)
    a[r][c] += b[r][c];
```

Answer: D

Question 25: Suppose that for a particular chip the multiplication operation is pipelined and takes three cycles to produce a value (the book would call this "latency 3, issue 1"). This would mean that computing `((a*b)*c)*d` would require 9 cycles. Using reassociation, what is the minimal number of cycles needed for that same product? (2 points)

A 7
B 8
C 4
D 9
E 6
F 3
G 5

Answer: A

Question 26: For our five-stage pipeline we had to implement prediction for conditional jumps in order to avoid extra stalls in the pipeline, but we did not need to do that for conditional moves.

If the following instructions were also conditional (e.g., if `halt` had conditional forms like `haltge` and `haltl`), which ones would need prediction to avoid stalls?

Select all that apply (2 points, -0.5 per error)

A `halt` impacts fetch
B `OPq` can delay decision until writeback
C `call` impacts fetch
D `popq` can delay decision until writeback
E `irmovq` can delay decision until writeback

Answer: A C

Pledge:

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

Your signature here