CS 3330 Final Exam Spring 2017

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

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.

.....

Information for questions 1–2
Consider 64KB 4-way set associative cache with 256 byte blocks.

Question 1 [2 pt]: (see above) Given an address `x` represented as an unsigned integer, which of the following C expressions will result in the cache tag for that address?

A `(x << 8) >> 14`
B `(x >> 8) & 0x3F`
C `x >> 8`
D `x >> 14`
E `(x >> 14) & 0x3F`
F none of the above

Answer: D

Question 2 [2 pt]: (see above) Given an address `x` represented as an unsigned integer, which of the following C expressions will result in the cache index for that address?

A `x >> 14`
B `(x >> 8) & 0x3F`
C `(x << 8) >> 14`
D `(x >> 14) & 0x3F`
E `x >> 8`
F none of the above

Answer: B

Information for questions 3–6

Consider the five-stage Y86 processor design we discussed in lecture. Suppose the stages take the following amounts of time (including register delays):

Stage	Time (ns)
Fetch	1.0 ns
Decode	0.8 ns
Execute	0.5 ns
Memory	1.2 ns
Writeback	0.8 ns

Question 3 [2 pt]: (see above) Suppose we execute a function which runs 1000 instructions on this processor. What is the **fastest** execution time the program can have?

A about 1.2 microseconds
B about 4.3 microseconds
C about 6 microseconds
D about 0.5 microseconds
E about 1 microsecond
F about 2.4 microseconds

Answer: A

Question 4 [2 pt]: (see above) Suppose we split the memory stage of this processor into two halves, each of which takes 0.6 ns. If the processor previously achieved around 2.0 cycles per instruction on some program and this change required 10% of instructions to stall for an additional cycle (beyond how much they previously stalled), then how would the throughput of the changed processor compare to the original processor? 2.4 ns/instr versus 2.1 ns/instr

A it would be about 0.2 ns/instruction faster
B it would be about 0.1 ns/instruction faster
C it would be about the same speed
D it would be about 0.1 ns/instruction slower
E it would be about 0.2 ns/instruction slower
F none of the above

Answer: F

Question 5 [2 pt]: (see above) Suppose our memory and fetch stages use a cache and, on a cache miss, these stages stall the pipeline for 10 cycles. Suppose in a given benchmark program with thousands of instructions, 1% of instructions trigger a cache miss stall in the fetch stage and 1% of instructions trigger a cache miss stall in the memory stage and there are *no other causes of stalls*. What is the throughput of the processor for this program in cycles per instruction?

A less than 1 cycle/instruction
B more than 1.3 cycles/instruction
C about 1.2 cycles/instruction
D about 1.1 cycles/instruction
E it depends on how many instructions stall both in fetch and in decode
F noe of the above

Answer: C

Question 6 [2 pt]: (see above) What is the cycle time of this processor in nanoseconds?

Answer: 1.2 ns

Information for questions 7–11

Suppose a processor has:

- 8KB pages
- 32-bit virtual addresses
- 28-bit physical addresses
- a two-level page table, with a 1KB page table at the first level, and 8KB page tables at the second level
- 4-byte page table entries
- a 16-entry 8-way set associative TLB
- in addition to the physical frame (page) number, page table entries contain a valid bit, a readable bit, a writeable bit, an executable bit, and a kernel-only bit.

Question 7 [2 pt]: (see above) Which of the following virtual addresses use the same TLB set as `0x12345678`? **Select all that apply.**

A `0x99345678`
B `0x123FD000`
C `0x00046678`
D `0x12345FFF`

Answer: A D

Question 8 [2 pt]: (see above) How many bits are in a virtual page number?

Answer: 19

Question 9 [2 pt]: (see above) How many bits of a page table entry are used? (Ignore any information about the location of pages on disk that might be stored in page table entries.)

Answer: 20 or 19 (think valid bit is information about the location of pages on disk)

Question 10 [2 pt]: (see above) Suppose this processor has a 32KB L1 cache whose tags are computed based on **physical** addresses. What is the *minimum* associativity that cache must have to allow the appropriate cache set to be accessed before computing the physical address that corresponds to a virtual address? If there is no associativity that will ensure this, write "none".

Answer: 4

Question 11 [2 pt]: (see above) Which of the following virtual addresses share the same second-level page table as `0x12345678`? **Select all that apply.**

A `0x123FD000`
B `0x00046678`
C `0x12345FFF`
D `0x99345678`

Answer: A C

Question 12 [2 pt]: `SIGUSR1` is a signal defined by the user. Consider the following C program, where `myprintf` is atomic (i.e., it can't be interrupted by signals and outputs immediately when called).

```
void handler1(int sig) {
    myprintf("Pirates\t");
    exit(0);
}

int main() {
    pid_t pid;
    signal(SIGUSR1, handler1);
    if((pid = fork()) == 0) {
        myprintf("of the\t");
        exit(0);
    }
    myprintf("Caribbean\t");
    kill(pid, SIGUSR1);
}
```

Which of the following outputs are **feasible** (possible)? **Select all that apply.**

A Pirates of the Caribbean
B Pirates Caribbean of the
C Caribbean of the Caribbean
D Caribbean of the Pirates

Answer: D

Information for questions 13–16
Consider the following two pieces of assembly

```
versionA:
    addq %rdx, %rax
    addq %rcx, %rax
    addq %rbx, %rax

versionB:
    addq %rdx, %rcx
    addq %rbx, %rax
    addq %rcx, %rax
```

Question 13 [2 pt]: (see above) On the five-stage pipelined processor we discussed in lecture, which of these would likely execute *faster*?

A Version A
B Version B
C They will have the same execution time.

Answer: C

Question 14 [2 pt]: (see above) Consider an out-of-order processor which takes two cycles to perform a 64-bit addition but can start a new addition every cycle. Which of the above assembly snippets would likely execute *faster on this out-of-order processor*?

A Version A
B Version B
C They will have the same execution time.

Answer: B

Question 15 [2 pt]: (see above) Consider an out-of-order processor which takes one cycle to perform a 64-bit addition but can start three additions every cycle. Which of the above assembly snippets would likely execute *faster on this out-of-order processor*?

A Version A
B Version B
C They will have the same execution time.

Answer: B

Question 16 [2 pt]: (see above) Suppose to speed up our five-stage pipelined processor we turned it into a six-stage pipelined processor by dividing the fetch stage into two fetch stages. Assume we made no changes to the processor that were not required because we divided the fetch stage. On this *changed processor*, which of the above assembly snippets would likely execute *faster*?

A Version A
B Version B
C They will have the same execution time.
D It depends on how we divided up the fetch stage.

Answer: C

Information for questions 17–19
Consider a 32-byte 2-way set associative cache with 4-byte cache blocks and an LRU replacement policy. Suppose a program using this cache performs 2-byte accesses of the following addresses (written in **decimal**) in the following order on an initially empty cache:
0, 16, 32, 2, 14, 34, 12, 20, 36.

Question 17 [2 pt]: (see above) How many **capacity** misses will the program experience?

Answer: 0

Question 18 [2 pt]: (see above) How many **compulsory** misses will the program experience?

Answer: 6 (for 0, 16, 32, 14, 20, 36)

Question 19 [2 pt]: (see above) How many **conflict** misses will the program experience?

Answer: 1 (for 2, same set as 0)

Question 20 [2 pt]: Suppose in our five stage pipeline processor is currently running instruction i_5 in its fetch stage, i_4 in decode, i_3 in execute, i_2 in memory, and i_1 in writeback. Which of the following will **always** happen before new values are stored in the condition code registers based on the result of instruction i_3? **Select all that apply.**

A The value of the predicted PC for the next cycle is stored in the predicted PC register
B i_5 is fetched from the instruction memory.
C If the instructions perform forwarding in this cycle, forwarded values are selected and output by MUXes
D If i_2 needs to write a value to memory, that value is stored in memory.

Answer: B C

Question 21 [2 pt]: Which of the following are part of a process's context? **Select all that apply.**

A The value of the `%rsp` register.
B The value of the valid bits in the L1 cache.
C The location of the exception table.
D The value of the page table base register.

Answer: A D

Question 22 [2 pt]: Suppose a 32K×8K matrix `A` with 1-byte elements is stored in row major order in virtual memory (i.e. `A[i][0]` and `A[i][1]` have consecutive virtual addresses). Assume only the program in question will occupy space in physical memory and the matrix is allocated from the beginning of a page and is initially located on disk.

```
for (i = 0; i < 32768; i++)
    for (j = 0; j < 8192; j++)
        A[i][j] = A[i][j] * A[i][j];
```

If the code fragment above yields 8K page faults, what is the size of a page in this architecture? (Hint: Due to demand paging, each time an access will touch a new page, it will generate a page fault.)

A 4 KB
B 8 KB
C 16 KB
D 32 KB

Answer: D

Question 23 [2 pt]: Your friend is designing the shared last-level (L3) cache of a Core i7-like processor (which consists of L1-I, L1-D, L2 and L3 caches and L1-ITLB, L1-DTLB and L2 TLB). The page size for the system is 4KB. The block size of the L3 cache they are considering is 128 bytes. Your friend would like to make the cache very large (64MB), but he is concerned that the synonym problem would complicate the design of the cache. What solution should you suggest to your friend?

A L3 does not have any synonym problem
B Make the L3 cache size smaller
C Implement a complex mechanism to locate the synonyms on a write
D Use separate L3-D cache and L3-I cache

Answer: A

Question 24 [2 pt]: Suppose you are learning the use of fork() with this example.

```
void myfork()
{
    printf("The\t");
    if (fork() == 0) {
        printf("Fullmetal\t");
        if (fork() == 0) {
            printf("Alchemist:\t");
        }
    }
    printf("Brotherhood\t");
}
```

Which of the following are **feasible** (possible)? (Assume that `printf` prints its output immediately.) **Select all that apply.**

A The Brotherhood Fullmetal Brotherhood Brotherhood Alchemist:
B The Fullmetal Brotherhood Brotherhood Alchemist: Brotherhood
C The Brotherhood Brotherhood Fullmetal Alchemist: Brotherhood
D The Brotherhood Fullmetal Alchemist: Brotherhood Brotherhood

Answer: B D

Question 25 [2 pt]: Our book advises against manipulating global variables or using `malloc` in signal handlers. What is a reason for this? **Select all that apply.** 0/1/2 points

A the signal handler must be very fast to make sure the program is responsive dropped — no credit/deduction for selecting
B the signal could be delivered while the program is manipulating global data structures 1 point for selecting
C the signal handler runs with less permissions than the rest of the process 1 point for not selecting

Answer: B

Information for questions 26–27
Consider storing a list of 100 four-byte integers being stored on a system with eight-byte pointers

Question 26 [2 pt]: (see above) Which of the following methods to store this list would take up the most space?

A a linked list
B a sentinel-terminated array
C a pointer to an array and size
D none of the above; multiple are tied for the most space

Answer: A

Question 27 [2 pt]: (see above) Suppose we want to extend this list to store 104 items. Assuming we **cannot** change the size of a memory allocation without copying it and we did not allocate extra space in advance, which of the following would require writing the most memory to do this?

A a pointer to an array and a size
B a sentinel-terminated array
C a linked list
D none of the above; multiple are tied for writing the most memory

Answer: (dropped) B

Question 28 [2 pt]: Conisder a processor with an 8-way set associative 32KB cache with 256 byte blocks with an LRU replacement policy. Which of the following addresses map to the same cache set as `0x12345678`? **Select all that apply.**

A `0x123FD078`
B `0x00000078`
C `0x12345600`
D `0x099996FF`

Answer: C D

Question 29 [2 pt]: When a program accesses a memory location that is currently swapped (AKA paged) to disk, this triggers a(n)

A fault
B interrupt
C abort
D trap
E thing that is not any kind of exception

Answer: A

Question 30 [2 pt]: Which of the following statements about loop unrolling are true? **Select all that apply.**

A Loop unrolling usually decreases program size.
B The processor will usually execute less instructions when running an unrolled loop.
C If the body of a loop accesses multiple pointers that might alias, the compiler cannot unroll a loop
D Loop unrolling usually improves locality in the instruction cache.

Answer: B

Question 31 [2 pt]: Given the C declaration `char *foo[100];` which of the following C expressions are true? **Select all that apply.**

A `sizeof(foo) == 100 * sizeof(char)`
B `sizeof(foo[0]) == sizeof(char *)`
C `sizeof(foo) == 100 * sizeof(char*)`
D `sizeof(&(*(foo))) == sizeof(foo)`

Answer: B C

Question 32 [2 pt]: After handling a page fault and successfully loading the page from disk, the kernel will

A start executing the interrupted program at the instruction that triggered the page fault
B restart the interrupted program from the beginning
C start executing the interrupted program at the instruction after the one that triggered the page fault
D crash the interrupted program

Answer: A

Question 33 [2 pt]: When the processor detects an exception, the processor generally jumps to a location specified

A by the kernel
B by the currently running program
C by the currently running program for asynchornous exceptions; by the kernel otherwise
D by the kernel for synchonous exceptions; by the currently running program otherwise

Answer: A

Question 34 [2 pt]: If a process A registers a signal handler for some signal and a process B sends that signal to that process, the signal handler normally

A executes in user mode in process B
B executes in user mode in process A
C executes in kernel mode
D doesn't execute at all unless process A specifically checks for signals

Answer: B

Information for questions 35–37

For these questions consider the following two functions. Assume that when these functions are called, `N` is a large positive integer.

```
int divisiblePairs_versionA(int *array, int N) {
    int count = 0;
    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j)
            if (array[i] % array[j] == 0)
                count += 1;
    return count;
}
int divisiblePairs_versionB(int *array, int N) {
    int count = 0;
    for (int ii = 0; i < N; ii += 8)
        for (int j = 0; j < N; ++j)
            for (int i = ii; i < ii + 8 && i < N; ++i)
                if (array[i] % array[j] == 0)
                    count += 1;
    return count;
}
```

Question 35 [2 pt]: (see above) Which version is likely to take up *more space in the instruction cache*?

A Version A
B Version B
C They will use the same or almost the same amount of space in the instruction cache.

Answer: B

Question 36 [2 pt]: (see above) Which version is likely to have *more cache misses*?

A Version A
B Version B
C They will have the same or almost the same number of cache misses.

Answer: A

Question 37 [2 pt]: (see above) Which version is likely to execute *significantly more instructions*?

A Version A
B Version B
C They will execute the same or almost the same number of instructions.

Answer: (dropped) B

Information for questions 38–39
Consider a system with 48-bit virtual addresses, 4KB pages, and a four-level page table where page tables at each level contain 512 8-byte page table entries.

Question 38 [2 pt]: (see above) Given a byte address `x` represented as an unsigned integer, which of the following C expressions will result in the virtual page number of the address?

A `x >> 12`
B `x >> 21`
C `(x >> 12) & 0xFFF`
D `(x >> 12) & 0x1FF`
E `(x >> 21) & 0x1FF`
F `(x >> 21) & 0xFFF`
G none of the above

Answer: A

Question 39 [2 pt]: (see above) Given a byte address `x` represented as an unsigned integer, which of the following C expresssions will result in the index of the second-level page table entry that would be accessed for this address?

A `(x >> 12) & 0x1FF`
B `x >> 12`
C `(x >> 30) & 0xFFF`
D `(x >> 30) & 0x1FF`
E `(x >> 12) & 0xFFF`
F `x >> 30`
G none of the above

Answer: D

Information for questions 40–42
For these questions, considering adding an instruction `jrreq` which conditionally jumps based on whether two registers are equal. For example

```
    jrreq %r10, %r11, Foo
After:
    subq %rcx, %rdx
    halt
Foo:
    addq %r8, %r9
```

would execute the `subq` if `%r10` and `%r11` were nonequal, and the `addq` if they were equal.

Question 40 [2 pt]: (see above) Suppose we are executing the following assembly on our five-stage pipelined processor modified to add this instruction where the `jrreq` is **not taken**:

```
     mrmovq 8(%r12), %r11
     jrreq %r11, %r10, After
     addq %rax, %r9
     subq %rbx, %r12
 After:
     xorq %r10, %r9
     rrmovq %r9, 8(%r12)
```

During the cycle before `addq` is fetched, what are the values of the bubble and stall signals in the pipeline? Use **S** to represent stall, **B** to represent bubble, and **N** to represent neither signal being set. .5 points per box

N	F	B	D	B	E	N	M	N	W

Question 41 [2 pt]: (see above) Consider executing the following assembly code on a five-stage pipelined processor with `jrreq` added:

```
     mrmovq 8(%r10), %r11
     jrreq %r11, %r10, After
```

Assume this processor uses the stages we discussed in class and implements forwarding whenever it would save clock cycles without dramatically increasing the cycle time. Will this snippet require forwarding or stalling on this processor when the branch is correctly predicted? **Select all that apply.** Ignore stalling parts; 0/.5/1/2 for 0/1/2/3 others correct

A It will require stalling for at least one cycle. dropped
B It will require stalling for at least two cycles. dropped
C It will require forwarding of `%r10`.
D It will require forwarding of `%r11`.
E It will require forwarding of a values other than `%r10` or `%r11`

Answer: A D

Question 42 [2 pt]: (see above) Which of the following are true about adding this instruction to our **single-cycle processor** design? **Select all that apply.** -1 for each wrong

A The instruction would require a longer instruction encoding than any other instruction. same as mrmovq
B It would require adding an additional 64-bit comparator to the processor. comparison can be done with existing ALU

Answer:

Information for questions 43–45
Consider the following two assembly snippets

```
snippetA:
    addq %r8, %rax
    mrmovq 8(%rax), %rbx
    mrmovq 16(%rax), %rcx
    addq %rbx, %rdx
    addq %rcx, %rdx

snippetB:
    mrmovq 8(%rax), %rbx
    addq %rbx, %rdx
    mrmovq 16(%rax), %rcx
    addq %rcx, %rdx
    addq %r8, %rax
```

Question 43 [2 pt]: (see above) On our five-stage pipelined processor, what is the difference between the number of cycles required to execute `snippetA` and `snippetB`?

A `snippetA` requires 1 less cycles
B `snippetA` requires the same number of cycles
C `snippetA` requires 1 more cycles
D `snippetA` requires 2 more cycles
E none of the above

Answer: E

Question 44 [2 pt]: (see above) Assuming an out-of-order processor has multiple copies of all arithmetic functional units and a cache that is capable of handling multiple accesses in parallel, which pairs of instructions could from `snippetA` could it execute at the same time? **Select all that apply.**

A `mrmovq 16(%rax), %rcx` and `addq %rbx, %rdx`
B `mrmovq 8(%rax), %rbx` and `mrmovq 16(%rax), %rcx`
C `addq %r8, %rax` and `mrmovq 8(%rax), %rbx`
D `addq %rbx, %rdx` and `addq %rcx, %rdx`

Answer: A B

Question 45 [2 pt]: (see above) Suppose we modified our five-stage pipelined processor by splitting the execute stage into two execute stages. Assume this modified processor requires operands for ALU operations near the beginning of the first execute stage and only makes results available near the end of the second execute stages. On this modified processor, how much more will `snippetA` stall compared to the unmodified processor?

A `snippetA` will stall the same amount
B `snippetA` will stall for 1 additional cycle
C `snippetA` will stall for 2 additional cycles
D `snippetA` will stall for 3 additional cycles
E none of the above

Answer: D

Pledge:

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

_____

Your signature here