CS 3330 Exam 3 Spring 2018

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.

. . . . . .

Question 1 [2 pt]: Given a `long *x` stored in register `%r8` and `long y` stored in register `%r9`, the C expression `x[y+1] += y` is equivalent to

A
```
addq %r9, 8(%r8,%r9,8)
```
B
```
leaq 8(%r8,%r9,1), %rax
addq %r9, (%rax)
```
C
```
addq %r9, 1(%r8,%r9,8)
```
D
```
leaq 8(%r8,%r9,8), %r9
```
E
```
leaq 1(%r8,%r9,8), %r9
```

Answer: A

Question 2 [2 pt]: If a page fault occurs, and the operating system responds to the page fault by allocating or swapping in the page whose invalidity triggered the fault, then the operating system will switch to user mode and resume executing the faulting program starting with _____.

A an instruction at an address specified in the exception table
B the instruction that triggered the fault
C the instruction immediately after the one that triggered the fault
D the first instruction in the function that triggered the fault
E none of the above; the OS will crash the program instead

Answer: B

Information for questions 3–4

Suppose a program accesses the following addresses in the following order, where each access is a one byte read or write:

- `0x0FF`
- `0xFFF`
- `0x100`
- `0x101`
- `0xFFE`
- `0xFF6`
- `0x0FF`

Question 3 [2 pt]: (see above) If the accesses above are to *virtual* addresses, how many TLB misses would a system with 256-byte pages and an initially empty, two-entry direct-mapped TLB experience? Write your answer as a base-10 number.

Answer: 4 (for 0FF, FFF, 100, FFE)

Question 4 [2 pt]: (see above) If the accesses above are to *physical* addresses, how many cache misses would a system with an initially empty 8 byte, two-way set associative cache with 2 byte blocks and a LRU replacement policy experience? (You may assume there are no other accesses.)

Answer: 5 (for 0FF, FFF, 100, FF7, 0FF (conflicts with FFF, FF6))

Question 5 [2 pt]: When the operating system switches from one process to another, which of the following will the operating system do? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [✓] change the value of `%rsp`

B [✓] change the page table base register

C [ ] overwrite one or more page table entry's valid bits

D [ ] change the exception table base register

Information for questions 6–7
Consider the following two Y86-64 assembly snippets: Version A:

```
mrmovq (%rax), %rax
addq %rbx, %rax
addq %rcx, %rdx
addq %rdx, %rax
```

and Version B:

```
mrmovq (%rax), %rax
addq %rbx, %rcx
addq %rcx, %rax
addq %rdx, %rax
```

Question 6 [2 pt]: (see above) On an out-of-order processor which has 10 pipelined ALUs that perform additions with two cycle latency and a load functional unit that takes 3 cycles to perform each load, version A will be _____ version B. Assume that no component of the processor other than these functional units significantly affects the relative performance.

A 2 or more cycles slower than
B 1 cycle slower than
C take the same number of cycles as
D 1 cycle faster than
E 2 or more cycles faster than

Answer: C

Question 7 [2 pt]: (see above) On a five-stage pipelined processor like the one described in our textbook with forwarding and branch prediction, version A will _____ version B.

A be 2 or more cycles slower than
B be 1 cycle slower than
C take the same number of cycles as
D be 1 cycle faster than
E be 2 or more cycles faster than

Answer: B

Information for questions 8–9
Consider the following function:

```
int foo(int *A, int *B, int N) {
    int total = 0;
    A[0] += B[0];
    for (int i = 1; i < N; ++i) {
        A[i] += A[i - 1] * B[i / 2];
        total += A[i];
    }
    return total;
}
```

Question 8 [2 pt]: (see above) The potential for the arrays pointed by `A` and `B` to overlap makes it more difficult for a compiler to perform which of the following optimizations on the above code? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] unrolling the loop

B [ ] using multiple accumulators for `total` don't need to reorder assembly to use mult. accums.

C [ ] keeping `A[i]` in a register between iterations in the loop so the access to `A[i - 1]` avoids the cache A[i] and A[i-1] can't overlap

D [✓] using vector instructions to implement the loop reorders +=s

Question 9 [2 pt]: (see above) Using vector instructions like the SSE1-4 instructions provided on x86-64 to optimize the above `for` loop would be simpler if _____. **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] `i < N` were replaced by `B[i] != 0` hard to tell how many vector elements to use

B [✓] `A[i - 1]` were replaced by `A[i]` allows A[i] to be computed without computing A[i-1] first

C [✓] `B[i / 2]` were replaced by `B[i]` easier to load B[i], B[i+1], etc. into vector than B[i/2], B[(i+1)/2], etc.

D [✓] `A[i - 1]` were replaced by `A[i + 1]` allows A[i] to be computed without computing A[i-1] first

Information for questions 10–12
Consider the following Y86-64 assembly snippet:

```
subq %r8, %r9
popq %rax
addq %rax, %r9
pushq %rbx
xorq %r8, %rbx
```

Question 10 [2 pt]: (see above) On an out of order processor with sufficiently many functional units and sufficiently large instruction queues, etc., which instructions could be executed at the same time as `subq %r8, %r9`? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [✓] `popq %rax`

B [ ] `addq %rax, %r9`

C [✓] `pushq %rbx`

D [✓] `xorq %r8, %rbx`

Question 11 [2 pt]: (see above) When the above assembly snippet is executing on the five-stage pipelined implementation described in our textbook with forwarding, which of the following forwarding operations will be performed? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [✓] the value of `%rsp` from `popq` to `pushq`

B [ ] the value of `%r8` from `subq` to `xorq`

C [ ] the value of `%rbx` from `pushq` to `xorq`

D [✓] the value of `%r9` from `subq` to `addq`

Question 12 [2 pt]: (see above) Consider the above assembly snippet executing on a pipelined implementation with forwarding and the following pipeline stages:

A. fetch and decode
B. execute
C. memory part 1
D. memory part 2
E. writeback

Assume that the split memory stage requires the memory address being accessed be available near the beginning of the memory part 1 stage and does not make its result available until near the end of the memory part 2 stage. If this processor fetches the `subq` instruction during cycle 1, during what cycle number will it perform the writeback stage of the `xorq` insturction?

Answer: 11

Information for questions 13–17

Consider a system with:

- two-level page tables
- 42-bit virtual addresses
- 40-bit physical addresses
- 8 byte page table entries
- 64KB (2^{16} byte) pages
- 64KB (2^{16} byte) page tables at each level
- a 32-entry sixteen way set-associative TLB

Question 13 [2 pt]: (see above) Accessing which of the following addresses will use the same TLB set as `0x00 123 456 789`? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☐ `0x00 123 444 444`

B ☑ `0x00 0A9 876 789`

C ☐ `0x00 000 006 789`

D ☐ `0x00 136 345 678`

Question 14 [2 pt]: (see above) Suppose a program has pages allocated (and marked as valid in its page table) at `0x00 000 100 000`, `0x00 000 200 000`, and `0x1F FFF F00 000`, and all of its other pages are marked as invalid in its page table. What is the minimum number of pages **of page tables** that must be allocated for the program? Write your answer at a base-10 integer.

Answer: 3

Question 15 [2 pt]: (see above) How many bits are in physical page numbers? Write your answer as a base-10 number.

Answer: 24

Question 16 [2 pt]: (see above) How many tag bits are stored alongside each TLB entry? Write your answer as a base-10 number.

Answer: 25

Question 17 [2 pt]: (see above) Which of the following addresses use the same **first-level** page table entry as `0x00 123 456 789`? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☑ `0x00 123 444 444`

B ☐ `0x00 0A9 876 789`

C ☑ `0x00 136 345 678`

D ☐ `0x00 000 006 789`

Information for questions 18–19

Suppose a system has:

- 2^{12} byte pages
- 32-bit virtual addresses
- single-level page tables
- 4-byte page table entries
- a fully associative sixteen-entry TLB
- a page table base register containing the physical byte address `0x10000`

Suppose a program running is user mode is trying to read from virtual address `0x208A3`.

Question 18 [2 pt]: (see above) If a TLB lookup is performed for this memory access, the TLB tag will be compared to what value? Write your answer as a *hexadecimal* number.

Answer: 0x20

Question 19 [2 pt]: (see above) If a TLB miss occurs, the MMU will read a page table entry from what address? Write your answer as a *hexadecimal* number or write "none" if the MMU will not read a page table entry in this case.

Answer: 0x10080 (half-credit for 0x10020)

Question 20 [2 pt]: Suppose program *A* tries to access a page of memory, but it is not loaded yet. As a result, the operating system starts loading it, and while it is loading runs program *B* until the page is loaded. Once the page is loaded, it immediately switches back to program *A*. How many exceptions must occur during this process? (Use our textbook's definition of exception and write your answer as a base-10 number.)

Answer: 2

Information for questions 21–22

Consider the following Y86-64 code:

```
irmovq $0x51, %r9
irmovq $0x3F, %r10
mrmovq 0x100(%r10), %r8
xorq %r9, %r10
cmovge %r8, %r9
```

Question 21 [2 pt]: (see above) What is the final value of `%r10` when this code is executed? Write your answer as a *hexadecimal* number.

Answer: 0x6E

Question 22 [2 pt]: (see above) When a **single-cycle** Y86 implementation (built like we described in lecture) is executing the `cmovge` instruction, one of the "register number to write" inputs of the register file will be equal to `0xF` (the constant for "no register") and the other will be equal to

A the register number for `%rsp` (the stack pointer)
B the value of the condition code `SF`
C `0xF`
D part of the instruction memory's 80-bit output the register # for %r9 in the instruction
E part of the ALU's 64-bit result

Answer: D

Question 23 [2 pt]: When a function A in object file X calls a function B in object file Y, the object file X references function B using

A the name of the object file Y and the name of function B within it
B the name of the object file Y and function B's location in that file
C the difference between the call's memory address and B's memory address
D the memory address of B's first instruction in executable, but not object files
E the name of the function B if linked with another version of B, will use it

Answer: E

Question 24 [2 pt]: When an exception occurs, the operating system or the processor usually will _____. (Use our textbook's definition of exception.) **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [✓] switch out of user mode (if the processor was in user mode)
B [ ] change the exception table base register
C [✓] save the values of registers which are caller-saved in the calling convention
D [✓] save the values of registers which are callee-saved in the calling convention

Question 25 [2 pt]: If an L1 cache has 20-bit tags and the TLB on the same system also has 20-bit tags, then for any particular address a program accesses, the two corresponding tags _____.

A are the same only if the page table is set up in a particular way
B are never the same
C are the same only if the TLB is fully associative
D are the same only if L1 cache is fully associative
E are always the same

Answer: A

Question 26 [2 pt]: In the five-stage pipelined processor design we discussed in lecture, the register number to write inputs to the register file (called `reg_dstE` and `reg_dstM` in HCLRS) are ***most directly*** connected to

A the data memory's output
B the pipeline registers between the memory and writeback stages
C the instruction memory's output
D the pipeline registers between the decode and execute stages
E the pipeline registers between the fetch and decode stages
F the condition code register output

Answer: B

Information for questions 27–29
Consider the following Y86-64 assembly code:

```
    irmovq $10000, %r9
    irmovq $10, %r8
    irmovq $1, %r10
    irmovq $0, %r11
loop:
    subq %r8, %r9
    jle after_loop
    addq %r10, %r11
    jmp loop
after_loop:
    halt
```

Question 27 [2 pt]: (see above) What is the value of `%r11` when the program halts? Write your answer as a base-10 number.

Answer: 999
(half-credit for 1000 or 1001 or 998)

Question 28 [2 pt]: (see above) Suppose this program is run on a five-stage pipelined processor with branch prediction and forwarding as discussed in our textbook. (Recall that this processor's branch prediction predicts all branches as taken and corrects mispredictions by fetching the correct instruction in the conditional jump's memory stage.) During the first iteration of the loop above, when the `subq` is completing its writeback stage, what instruction is being fetched?

A `jmp loop`
B `subq %r8, %r9`
C `halt`
D `jle after_loop`
E `addq %r10, %r11`
F none of the above

Answer: E

Question 29 [2 pt]: (see above) How many **more cycles** does this program take to execute on the five-stage processor with branch prediction and forwarding discussed in our textbook compared to a single-cycle processor. Write your answer as a base-10 decimal number. extra 2 cycles/loop (branch mispredict) + 4 cycles at end

Answer: 2002
(-0.5 for off-by one or two; half-credit for 1990-2010)

Question 30 [2 pt]: Consider the following loop, where `I`, `J`, `N`, and `M` are integer constants. Assume `N` and `M` have large values

```
for (int ii = 0; ii < N; ii += I)
    for (int jj = 0; jj < M; jj += J)
        for (int i = ii; i < ii + I; ++i)
            for (int j = jj; j < jj + J; ++j)
                A[i * N + j] += B[j * N + i];
```

How should the values of `I` and `J` be chosen to achieve best performance?

A based primarily on the cache size
B based primarily on how many elements of `A` and `B` fit into registers
C to be equal to N and M, so the compiler can eliminate the outermost loops
D to be one, so the compiler can eliminate the innermost loops
E based primarily on the size of vector registers the processor supports

Answer: A

Information for questions 31–32
Suppose one builds a Y86-64 processor from components which take the following amounts of time to perform operations (reads, writes, calculations, etc.):

- instruction memory — 100 ps
- data memory – 150 ps
- ALU – 50 ps
- register file read – 150 ps
- register file write – 150 ps
- PC increment based on icode – 75 ps

Question 31 [2 pt]: (see above) In a single-cycle implementation of this processor, what is the minimum possible cycle time, excluding any register delay for the PC register? (You may assume any component whose time wasn't mentioned above requires negiligible time.) Write your answer as a base-10 number of picoseconds. critical path: imem to reg file read to ALU to dmem to reg file write

Answer: 600 ps (half-credit for 675 ps)

Question 32 [2 pt]: (see above) In a five-stage implementation of this processor, using the design we discussed in lecture, what is the minimum possible cycle time assuming pipeline registers and the PC have a 10 ps register delay? (You may assume any component whose time wasn't mentioned above requires negiligible time.) Write your answer as a base-10 number of picoseconds. 100 + 75 ps for fetch (imem + PC increment) + 10 ps register delay

Answer: 185 ps (half-credit for 175/160 ps)

Information for questions 33–35
Consider the following C loops, given that `N` is a large integer constant:

```
// Version A:
for (int i = 0; i < N; ++i)
    for (int j = 0; j < N; ++j)
        for (int k = 0; k < N; ++k)
           A[i * N + j] += B[j * N + k];

// Version B:
for (int j = 0; j < N; ++j)
    for (int i = 0; i < N; ++i)
        for (int k = 0; k < N; ++k)
           A[i * N + j] += B[j * N + k];

// Version C:
for (int k = 0; k < N; ++k)
    for (int j = 0; j < N; ++j)
        for (int i = 0; i < N; ++i)
           A[i * N + j] += B[j * N + k];
```

Question 33 [1 pt]: (see above) Which version has the best temporal locality in accesses to `B`?

A version A
B version B
C version C
D multiple are tied

Answer: C

Question 34 [2 pt]: (see above) An efficient cached block version of the above loops would access adjacent elements of A without accessing other, further away elements of A in between _____.

A more often than at least one version of the loop above and less often than at least one verison of the loop above
B less often than any version of the loop above
C more often than any version of the loop above

Answer: A

Question 35 [1 pt]: (see above) Which version has the best spatial locality in accesses to `A`?

A version A
B version B
C version C
D multiple are tied for best

Answer: A

Question 36 [2 pt]: On a system with a three-level page table, when the program accesses memory at some address, the MMU (memory management unit) only accesses the second-level page table entry for that address _____. **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] if the third-level page table entry for that address is valid

B [✓] if the first-level page table entry for that address is valid

C [✓] on a TLB miss

D [ ] when the processor is in kernel mode

Question 37 [2 pt]: Suppose a system has 34-bit virtual addresses, 38-bit physical addresses, 3-level page tables, and 4-byte page table entries. If every page table on this system is exactly one page (with no unused space), then how many page offset bits do virtual addresses on this system have? Write your answer as a base-10 number.

Answer: 10

Question 38 [2 pt]: Writing to a virtual page whose page table entry's *writeable* bit is false while in user mode will _____. (Use our textbook's terminology for exception-related terms.)

A stall the processor until the page is loaded into cache
B do different things depending on whether the data for the address being written is stored in the L1 cache
C trigger some kind of fault
D cause the write to be silently ignored
E none of the above; virtual pages don't have page table entries

Answer: C

Question 39 [2 pt]: When a program performs a system call, the operating system performs the requested service, then usually switches to user mode and resumes executing the faulting program starting with _____.

A the first instruction in the function that triggered the system call
B the instruction immediately after the one that triggered the system call
C an instruction at an address specified in the exception table
D the instruction that triggered the system call
E none of the above; the OS will crash the program instead

Answer: B

Information for questions 40–41

Consider a 4-way set associative 256KB (2^{18} byte) cache with 64-byte blocks and a write-through, no-write-allocate policy, and a pseudo-LRU (least recently used) replacement policy.

Question 40 [2 pt]: (see above) On a system with 32-bit addresses, how many bits *of each set* are devoted to storing cache tags? Write your answer as a base-10 number.

Answer: 64 (half-credit for 16)

Question 41 [2 pt]: (see above) Data for which of the following addresses in this cache could be stored in the same set as data for address `0x123456`? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] `0xFFB456`

B [✓] `0x12344F`

C [ ] `0x123481`

D [✓] `0x463456`

Question 42 [2 pt]: Increasing the number of pipeline stages in a processor generally **increases** _____. **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] the cycle time of the processor

B [✓] the portion of the cycle time devoted to pipeline register delays

C [✓] the mean cycles per instruction (CPI)

D [✓] the time from when a conditional jump is fetched until when it is determined whether or not the jump is taken

Information for questions 43–44
Consider the following C code:

```
unsigned char foo[N];
...
for (int j = 0; j < 1024; ++j) {
    for (int i = j; i < N; i += 1024) {
        foo[i] += 1;
    }
}
```

Assume that the loop is compiled so that only the array `foo` is kept in memory (all other values are kept in registers), and memory accesses in the loop to `foo` are not omitted or reordered.

Suppose this code is running on a system with a 16KB (2^{14} byte) two-way set associative data cache with 256-byte (2^8) blocks with an LRU replacement policy and a write-back, write-allocate policy, and that the cache is initially empty when the outer for loop first starts.

Question 43 [2 pt]: (see above) If `N` is `(16 * 1024 + 2)`, then how many data cache misses will the above loop experience? Write your answer as a base-10 number.

Answer: 70 (64 + 6) or 71

Question 44 [2 pt]: (see above) If `N` is `(16 * 1024)`, then how many data cache misses will the above loop experience? Write your answer as a base-10 number.

Answer: 64 or 65

Question 45 [2 pt]: Which of the following is likely to be part of a page table entry? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [ ] a page offset

B [ ] a virtual page number

C [✓] a physical page number

D [✓] a valid bit

Question 46 [2 pt]: Which of the following operations are typically triggered by an instruction executed by the operating system in kernel mode? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [✓] invalidating a TLB entry when a page table entry changes

B [ ] invalidating an L1 data cache block when its value is changed in memory

C [✓] switching from kernel mode to user mode

D [✓] setting the values in the exception table

Information for questions 47–48
Consider the array defined using the following C code:

```
int array[5] = {0x123456,0x234567,0xDDDD8,0xEEEE8,0xFFFF8};
```

Suppose the array above is defined on a little-endian system where `int`s are four bytes where the first byte of the array is stored at memory address `0x70000`.

Question 47 [2 pt]: (see above) What is the value of the sixth **byte** of array (i.e. the byte at address `0x70005`)? Write your answer as a *hexadecimal* number.

Answer: 0x45

Question 48 [2 pt]: (see above) What is the result of `(*(array + 2)) & 0x37`? Write your answer as a *hexadecimal* number.

Answer: 0x10

Pledge:

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

_____

Your signature here