CS 3330 Exam 3 Fall 2017

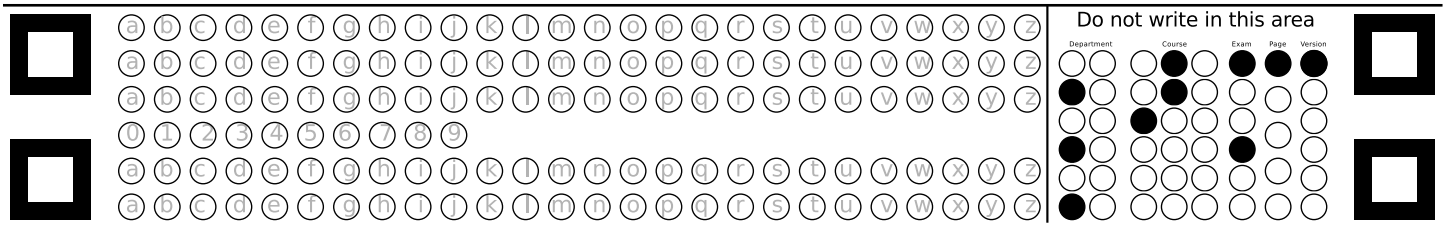
Name: _____ Computing ID: _____

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.

. . . . . .



Information for questions 1–2
Consider a system with 32-bit virtual addresses, 1KB (2^{10} byte) pages, and 34-bit physical addresses. Suppose this system uses 4-byte page table entries, with the following fields, from most significant to least significant bit (when the page table entry is interpreted as a 4-byte integer):

- a 24-bit physical page number
- 1 write-allowed bit
- 1 execute-allowed bit
- 1 kernel-only bit
- 4 unused bits
- 1 valid bit

Question 1 [2 pt]: (see above) If an `unsigned int addr` contains a virtual address on this system, which of the following C expressions would be equal to the page offset of that address? (Recall that `>>` does a logical shift on unsigned values.)

A `(addr >> 10) & 0xFFF`
B `addr >> 34`
C `addr >> 8`
D `addr & 0x3FF`
E `addr >> 10`
F `addr & 0xFF`
G none of the above

Answer:

Question 2 [2 pt]: (see above) If we have loaded a page table entry into an `unsigned int pte`, then which of the following C expressions would extract the physical page number from it? (Recall that `>>` does a logical shift on unsigned values.)

A `(pte >> 8) ^ 0xFFFFFF`
B `pte >> 10`
C `pte >> 8`
D `pte >> 24`
E `pte & 0xFFFFFF`
F `(pte & 0xFFFFFF) >> 8`
G none of the above

Answer:

Question 3 [2 pt]: Which of the following are true about kernel mode? Place a ✓next to each true answer. Leave other boxes blank.

A ☐ kernel mode can be entered via a function call

B ☐ code running in kernel mode may have its own dedicated regions of the address space

C ☐ code running in kernel mode can directly access I/O devices

D ☐ kernel mode can be entered via an interrupt

Question 4 [2 pt]: Which of the following are ways using **multiple accumulators** is likely improve performance? Place a ✓next to each correct answer. Leave other boxes blank.

A ☐ by decreasing the number of instructions a program executes

B ☐ by decreasing the number of pipeline stalls required due to data dependencies

C ☐ by allowing more instructions to executed in parallel with each other

D ☐ by making it easier for the processor to correctly predict the branches of a loop

Question 5 [2 pt]: On a system that implements swapping, when a program runs a `mov` instruction that tries to access a memory location that has not been loaded from disk, which of the following may happen? Place a ✓next to each correct answer. Leave other boxes blank.

A ☐ the processor looks up the location of the page fault handler in the exception table

B ☐ the operating system can context switch to another program while the disk read is happening

C ☐ the memory management unit of the processor tells the disk to start reading

D ☐ the operating system eventually restarts the program starting from the instruction **after** the `mov`

Question 6 [2 pt]: Which of the following statements about loop unrolling **are true**? Place a ✓next to each true statement. Leave other boxes blank.

A ☐ too much loop unrolling is likely to cause register spilling, decreasing overall performance

B ☐ loop unrolling can be combined with cache blocking

C ☐ loop unrolling will not improve performance on a single-cycle processor since there is no way for multiple instructions to execute in parallel

D ☐ compilers have trouble performing loop unrolling themselves because it requires analyzing code from multiple source files

Information for questions 7–10

Consider a system with:

- 64 KB (2^{16} byte) pages
- 28-bit physical page numbers
- 8 byte page table entries
- 36-bit virtual addresses
- two-level page tables, where first-level page tables are 4KB, and second-level tables are 16KB
- a 3-way, 384-entry TLB

Question 7 [2 pt]: (see above) When the processor is accessing the virtual address `0x012345678`, it will access the same TLB set as when it accesses _____. Place a ✓next to each correct answer. Leave other boxes blank. .

A ☐ `0x5FFB45678`

B ☐ `0x012FF5678`

C ☐ `0x012340000`

D ☐ `0x000005678`

Question 8 [2 pt]: (see above) Suppose the page table base register contains the byte address `0x10000`. What is the physical address of the first-level page table entry for `0x009231234`? Write your answer as a hexadecimal address.

Answer:

Question 9 [2 pt]: (see above) How much information (including metadata) is stored in the TLB?

A more than 21MB
B between 1KB and 6KB
C between 384 bytes and 1KB
D between 6KB and 64KB
E between 64KB and 21MB

Answer:

Question 10 [2 pt]: (see above) What is the size (in bits) of a physical address on this system? Write your answer as a base-10 number.

Answer:

Question 11 [2 pt]: Consider a five-stage pipelined processor like the one described in lecture and the textbook. Including register delays, each of the stages requires the following amount of time to perform its computations:

- fetch — 100 picoseconds
- decode — 50 picoseconds
- execute — 150 picoseconds
- memory — 200 picoseconds
- writeback — 50 picoseconds

How many picoseconds after a program starts fetching its first instruction will it finish the writeback stage of its third instruction? Assume the pipeline does not need to stall. Write your answer as a base-10 number of picoseconds.

Answer:

Question 12 [2 pt]: Consider the following C code:

```
int x[3] = {0x1234, 0x5678, 0xABCD};
```

If `int`s are four bytes and stored in little endian order, then the sixth byte of the array `x` is:

A `0xCD`
B `0x00`
C `0x56`
D `0x78`
E `0xAB`
F there is not enough information to answer
G none of the above

Answer:

Information for questions 13–14

Suppose an out-of-order processor has the following execution units (also known as functional units):

- four pipelined multipliers with four cycle latency
- four pipelined adders with two cycle latency

The following questions ask about the amount of time required perform the computations specified by an assembly snippet with these execution units, ignoring any overheads of fetching, decoding, etc. the instructions involved. Assume the overhead of forwarding values between functional units and any other costs to coordinating the computations are negligible.

Question 13 [2 pt]: (see above) Consider the following x86-64 assembly snippet:

```
mulq %r8, %r9
mulq %r9, %rax
mulq %r8, %rbx
```

How long will the computations in this assembly snippet take to perform? Write your answer as a base-10 number of cycles.

Answer:

Question 14 [2 pt]: (see above) Consider the following x86-64 assembly snippet:

```
mulq %r8, %r9
mulq %r10, %r11
addq %r9, %r11
```

How long will the computations in this assembly snippet take to perform? Write your answer as a base-10 number of cycles.

Answer:

Question 15 [2 pt]: Which of the following are likely to result in a **fault**? (Consider the textbook's terminology for types of exceptions.) Place a ✓next to each true answer. Leave other boxes blank.

A ☐ dividing by zero

B ☐ attempting to execute an invalid instruction

C ☐ accessing a memory location not in the process's address space

D ☐ the user pressing control-C

Question 16 [2 pt]: In the single-cycle processor design described in our textbook, when the processor executes the `ret` instruction, the data memory output is

A used to set the program counter
B used to restore saved registers
C used to compute the new stack pointer value
D not used
E used to read the fields of the `ret` instruction itself

Answer:

Information for questions 17–18
Consider a system with 16 KB (2^{14} byte) direct-mapped data cache with 64-byte blocks. In each of the below C code snippets, assume that:

- `array` is stored at an address that is a multiple of 2^{30}
- data caches are initially empty when starting the outer `for` loop
- only accesses to `array` use the data cache
- $512 = 2^9$, $128 = 2^7$, $1024 = 2^{10}$, $64 = 2^6$

Question 17 [2 pt]: (see above) Consider the following C code:

```
unsigned char array[1024 * 64];
...
for (int j = 0; j < 10; j++)
    for (int i = 0; i < 128; i++)
        array[i * 512] += 1;
```

How many data cache misses will the above nested loops experience on this system?

A $81920 = 10 \times 128 \times 64$
B 128
C $1280 = 10 \times 128$
D $416 = 10 \times 32 + 96 = 10 \times 128/4 + 3 \times 128/4$
E $992 = 32 + 10 \times 96 = 128/4 + 10 \times 3 \times 128/4$
F none of the above

Answer:

Question 18 [2 pt]: (see above) Consider the following C code:

```
unsigned char array[1024 * 64];
...
for (int j = 0; j < 10; j++)
    for (int i = 0; i < 128; i++)
        array[i * 512] += array[i * 128 + 64];
```

How many data cache misses will the above nested loops experience on this system?

A $163840 = 2 \times 10 \times 128 \times 64$
B $256 = 2 \times 128$
C $2240 = 10 \times 128 \times 2 - 10 \times 128/4$
D $2560 = 10 \times 128 \times 2$
E $1408 = 10 \times 128 + 128$
F none of the above

Answer:

Question 19 [2 pt]: Both signals and exceptions _____. Place a ✓next to each true answer. Leave other boxes blank.

A ☐ are features of the processor, primarily implemented in hardware

B ☐ must be used in order to access functionality that requires more privileges than normal user programs are allowed

C ☐ can be triggered by events external to the currently running program

D ☐ require something outside the signal or exception handler code to save the old program counter

Question 20 [2 pt]: Which of the folllowing best describes the locality for this code fragment?

```
for (i = 0; i < 1000; i++)
    for (j = 0; j < 1000; j++)
        b[j * 1000 + i] = a[i * 1000 + j];
```

A poor spatial locality and temporal locality for a
B good spatial locality for a and poor temporal locality for a and b
C good spatial locality for a and good temporal locality for b
D poor spatial locality and temporal locality for both a and b
E good spatial locality for b and poor temporal locality for a and b
F good spatial locality for b and good temporal locality for b

Answer:

Question 21 [2 pt]: Which of the following statements are true about vector instructions like the SSE extensions the x86-64 instruction set? Place a ✓next to each true statement. Leave other boxes blank.

A ☐ code with more data dependencies is more likely to benefit from vector instructions

B ☐ implementing vector instructions complicates the control path of the processor more than implementing out-of-order execution

C ☐ code with more conditionals is more likely to benefit from vector instructions

D ☐ code with more loops over large arrays is more likely to benefit from vector instructions

Question 22 [2 pt]: Which of the following changes to a cache design would **decrease** the number of set index bits it has? Place a ✓next to each change that would decrease the number of set index bits. Leave other boxes blank.

A ☐ doubling the total cache size and the associativity without changing the block size

B ☐ doubling the associativity without changing the block size or total cache size

C ☐ doubling the block size without changing the associatity or total cache size

D ☐ changing from a random replacement policy to an LRU replacement policy

Information for questions 23–24
Suppose a processor has 32-bit addresses and a 32KB 4-way set associative unified L1 cache with 64B blocks; a write-allocate, write-back policy; and a random replacement policy.

Question 23 [2 pt]: (see above) How many block offset bits does this cache use? Write your answer as a base-10 number.

Answer:

Question 24 [2 pt]: (see above) How many set index bits does this cache use? Write your answer as a base-10 number.

Answer:

Question 25 [2 pt]: Which of the following features are **more characteristic of a CISC** (complex instruction set computer) instruction set architecture than a RISC (reduced instruction set computer) architecture? Place a ✓next to each true answer. Leave other boxes blank.

A ☐ more addressing modes

B ☐ an instruction that increments a value in memory

C ☐ simpler instructions

D ☐ variable width instructions

Question 26 [2 pt]: Suppose we wanted to add a new add instruction `addq3` to Y86-64 that took three arguments, two source registers and one destination register. For example,

```
addq3 %rax, %rbx, %rcx
```

would add `%rax` and `%rbx` and store the result in `%rcx`. Which of the following would be true about adding this instruction to our single-cycle Y86-64 processor? Place a ✓next to each true answer. Leave other boxes blank.

A ☐ `addq3` would require a register number to be placed in a different part of the machine code than register numbers are placed for any other Y86-64 instruction

B ☐ Implementing this instruction would require modifying the register file to support more inputs and/or outputs.

C ☐ While this instruction is executing, one of the data inputs to the register file would be equal to the data memory output.

D ☐ While this instruction is executing, one of the register number inputs to the register file would be equal to the ALU output.

Question 27 [2 pt]: Consider the following three loops:

```
// Loop A
    for (int i = 0; i < 128; ++i) {
        A[i] = (B[i] + 10) * (C[i] - D[i]);
    }

// Loop B
    for (int i = 1; i < 128; ++i) {
        A[i] = A[i-1] - A[i+1];
    }

// Loop C
    for (int i = 0; A[i] != 0; ++i) {
        A[i] += B[i * 10];
    }
```

Which should be easiest to implement and improve the performance of with vector instructions like the SSE extensions to the x86-64 instruction set?

A Loop A
B Loop B
C Loop C
D none of the above; none can benefit from vector instructions

Answer:

Question 28 [2 pt]: Which of the following optimizations are likely to **increase** the number of L1 instruction cache and/or instruction TLB misses a program experiences?

A ☐ removing redundant calculations from a loop

B ☐ loop unrolling

C ☐ function inlining

D ☐ cache blocking

Question 29 [2 pt]: Suppose a processor lacked an explicit system call instruction. Which of the following would be an acceptable substitute for this instruction, allowing the operating system running on that processor to provide the same functionality without giving all programs unlimited access to the machine?

A have programs include the entire operating system code as a library and make a normal function call

B have the operating system use a signal handler in place of the system call trap handler

C have programs write the system call arguments to a file instead of using the system call instruction

D have programs perform a deliberate access to an invalid virtual page which is not used for anything else in place of the system call instruction

Answer:

Information for questions 30–32

For these questions, consider the following Y86-64 assembly snippet, running on the five-stage pipelined processor with forwarding and branch prediction discussed in lecture and our textbook.

```
    addq %rbx, %rax
    jle after
    rrmovq %rax, %rdx
after:
    subq %rax, %rdx
    xorq %rbx, %rdx
```

Recall that this processor implements branch prediction, in which it predicts all branches as taken, and on a misprediction, fetches the correct instruction when the conditional jump instruction executes its memory stage.

Question 30 [2 pt]: (see above) If **the `jle` is not taken**, which of the following forwarding operations are needed to execute this snippet correctly?

A ☐ `%rax` from `addq` to `rrmovq`

B ☐ `%rax` from `addq` to `subq`

C ☐ `%rdx` from `rrmovq` to `subq`

D ☐ `%rdx` from `rrmovq` to `xorq`

Question 31 [2 pt]: (see above) If **the `jle` is taken** and the `addq` instruction performs its fetch stage in cycle `0`, during what cycle will the `xorq` instruction perform its writeback stage? Write your answer as a base-10 number.

Answer:

Question 32 [2 pt]: (see above) If **the `jle` is not taken** and the `addq` instruction performs its fetch stage in cycle `0`, during what cycle will the `xorq` instruction perform its writeback stage? Write your answer as a base-10 number.

Answer:

Question 33 [2 pt]: In the single-cycle processor design described in our textbook, when the processor is executing the `pushq` instruction, the value of the output of the ALU is

A not used
B equal to the value of one of the register number inputs to the register file
C equal to the value of one the data inputs to the register file
D equal to the register number for `%rsp`
E none of the above

Answer:

Information for questions 34–36

For these questions, consider the following Y86-64 assembly snippet:

```
irmovq $0x4000, %rbx
irmovq $0x50, %rdx
mrmovq 0x3(%rbx), %rcx
subq %rcx, %rbx
addq %rdx, %rbx
```

Question 34 [2 pt]: (see above) On the five-stage pipelined processor design we discussed in class and in lecture with stalling and forwarding, how many cycles of stalling will be necessary to execute this assembly snippet? Write your answer as a base-10 number.

Answer:

Question 35 [2 pt]: (see above) Suppose we made a three-stage pipelined processor using the same components we used for our five-stage pipelined processor, such that the stages are:

1. Fetch and Decode
2. Execute
3. Memory and Writeback

Assume this processor implements all forwarding paths that will not require dramatically increasing the cycle time. In this processor, how many cycles of stalling will be required to execute the assembly snippet? Write your answer as a base-10 number.

Answer:

Question 36 [2 pt]: (see above) Suppose the three-stage pipelined processor (from the previous question) did not implement forwarding. How many cycles of stalling would be required to execute the assembly snippet then? Write your answer as a base-10 number.

Answer:

Question 37 [2 pt]: When an exception occurs, the *processor* _____. Place a ✓next to each correct answer. Leave other boxes blank.

A ☐ jumps to an address specified by the operating system

B ☐ starts running in kernel mode

C ☐ performs a context switch to another program

D ☐ writes the current program counter to the exception table

Question 38 [2 pt]: Suppose a processor has 32-bit virtual addresses, 36-bit physical addresses, and 8KB pages. If the processor uses a single-level page table with 4 byte page table entries, then how large is a page table on this processor?

A 2^{25} bytes
B 2^{13} bytes
C 2^{21} bytes
D 2^{23} bytes
E 2^{19} bytes
F it depends on what memory a program has allocated
G none of the above

Answer:

Question 39 [2 pt]: Which of the following are part of a process's *context* that an operating system stores during a context switch? Place a ✓next to each correct answer. Leave other boxes blank.

A ☐ the value of the page table base register
B ☐ the value of the program counter
C ☐ the value of the condition code SF
D ☐ the value of the exception table base register

Question 40 [2 pt]: Suppose a system has 4KB pages (2^{12} bytes), 39-bit virtual addresses, and three-level page tables, where page tables at each level are one page, containing 512 entries, and each page table entry is 8 bytes. If a program on this system has 3 MB ($2^{21} + 2^{20}$ bytes) of memory that it can access without triggering a fault, what is the minimum **size of its page tables**?

A 2 pages ($2 \cdot 2^{12}$ bytes) or less
B 3 pages or $3 \cdot 2^{12}$ bytes
C 4 pages or $4 \cdot 2^{12}$ bytes
D 5 pages or $5 \cdot 2^{12}$ bytes
E 6 pages or $6 \cdot 2^{12}$ bytes
F 7 pages or $7 \cdot 2^{12}$ bytes
G more than 7 pages or $7 \cdot 2^{12}$ bytes

Answer:

Question 41 [2 pt]: Suppose a processor has 16KB (2^{14} byte) pages, 32-bit virtual addresses and 24-bit physical addresses. Which L1 cache design would allow the L1 cache access to start before the TLB accesses finishes, but still allow the L1 cache to ensure that a given physical address always maps to the same cache set? Place a ✓next to each correct answer. Leave other boxes blank.

A ☐ a 128 KB (2^{17} byte), 4-way set associative cache with a write-through policy and 32 byte blocks

B ☐ a 16 KB (2^{14} byte) direct-mapped cache with a write-back policy and 64 byte blocks

C ☐ a 4 KB (2^{12} byte) direct-mapped cache with a write-through policy and 16 byte blocks

D ☐ a 96 KB ($3 \cdot 2^{15}$ byte), 6-way set associative cache with a write-back policy and 128 byte blocks and an LRU replacement policy

Question 42 [2 pt]: Consider the following C code:

```
int x[8] = {1, 2, 3, 4, 5, 6, 7, 8};
int *ptr;
ptr = x;
ptr += *ptr;
*ptr += *ptr;
ptr += *ptr;
```

What is the value of `*ptr` after this code executes?

A 4
B unknown; the code invokes undefined behavior
C 5
D 7
E 3
F 6
G none of the above

Answer:

Question 43 [2 pt]: Suppose the variable `a` is declared as an `long*` and stored in the register `%rax` and the variable `b` is declared as an `long*` and stored in the register `%rbx`. Which of the following is a correct translation of the C code `*a += *b;` to x86-64 assembly?

A
```
leaq 8(%rbx,%rax), %rcx
movq %rcx, (%rbx)
```

B
```
leaq (%rax,8), %rbx
```

C
```
movq (%rax), %rcx
addq %rcx, %rbx
movq (%rbx), %rbx
```

D
```
movq (%rax), %rcx
addq %rcx, (%rbx)
```

E none of the above

Answer:

Pledge:

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

_____

Your signature here