Computer Science 605.411

Final Exam

1. A certain program is executed on a multi-core system and consists of two parts: Part1 and Part2. Part1 is a single task that is purely sequential and can only be performed by a single processor core. Part2 consists of N independent tasks. Each task in Part2 takes half the time required for the single task in Part1 and must be performed by a single processor core.

a) (5) If the number of cores is 16 and N (the number of independent tasks in Part2) is 32, what is the greatest percentage of the potential speedup that can be provided by the 16-core system compared to a using only one core for the entire program? Recall that the potential speedup is just the number of cores in the multi-core system.

Assuming a task in part 1 takes 2 units of time and a task in part 2 takes 1 unit of time
Potential speedup is 16 (for the 16 cores)
The time for the process to execute on one core is 32 + 2 = 34
The time for the process to execute on 16 cores is 2 + (32 / 16) = 4
34 / 4 is a speedup of 8.5
The percentage of potential speedup is 8.5 / 16 = 53.125%

b) (8) For this 16-core system to achieve 80% of its potential speedup, how many independent tasks (each of which takes half the time required for the single task in Part 1) must Part2 contain? Your answer should be a whole number since there are no partial tasks.

$$\% \, Speedup = \frac{\frac{N+2}{ceiling(2+\frac{N}{16})}}{16}$$

ceiling() meaning round up (1.5 -> 2) since you cant run 24 processes on 16 cores in 1.5 units of time for this system, it would take 2 units of time
N = 126 independent tasks
(Note that there are multiple solutions due to the ceiling() function, including 139)
(Checking)

$$\frac{\frac{126+2}{ceiling(2+\frac{126}{16})}}{16} = 80\% \, speedup$$

c) (2) Is increasing the number of tasks in Part2 an example of weak scaling or an example of strong scaling?

Strong scaling is defined as how the solution time varies with the number of processors for a fixed total problem size. This would not be an example of strong scaling since we are increasing the problem size, not the number of processors. Weak scaling is defined as how the solution time changes with the number of processors for a fixed problem size per processor. This more closely matches the example in part b than strong scaling, thus this is a better example of weak scaling.

2. (5) A computer system has a byte addressable data memory and uses 32-bit physical addresses. Its 4-way set associative data cache employs 256-byte lines and contains a total of 1024 sets. Assume that the only information needed to manage the data cache consists of the bits for the MESI protocol, the tags and the pseudo-LRU bits. If the cache management data is stored in a separate data structure (the cache directory), what is the minimum number of bytes (not bits) required for the directory for the entire cache?

Set bits = $\log_2(1024)$ = 10 set bits
Offset bits = $\log_2(256)$ = 8 offset bits
Tag bits = 32 – 10 – 8 = 14 tag bits per set
Need 14 bit tags for each set -> 14 * 1024 = 14336 bits
Need 2 bits per set to store MESI protocol info -> 1024 * 2 = 2048 bits
For a 4-way set associative cache we need 3 psuedo-LRU bits per line -> 256 * 3 = 768 bits
(2048 + 14336 + 768) / 8 = 2144 total bytes

3. A superscalar version of our MIPS system contains two degree-4 superpipelines (pipe1 and pipe2). Each pipeline has 5 stages and runs at a 2.5 GHz clock rate. Assuming there are no stalls, answer the following questions:

a) (2) What is the peak MIPS rating for this superscalar system?

MIPS rating=clock_rate/(10^6*CPIavg)
1 superpipelined system can execute 1 instruction per clock cycle, while a superscalar system with two superpipelines can execute 2 instructions per clock cycle, thus the peak CPI for the system is 0.5
2.5 * 10 ^ 9 / (10^6 * 0.5) = 5000

b) (2) What is the minimum time interval in nano-seconds between the completion of any two instructions on pipe1?

Interval = 1 / clock cycle frequency = 4 * 10 ^-10 s = 0.4 ns

c) (2) What is the minimum time interval in nano-seconds between the completion of an instruction on pipe1 and the completion of an instruction on pipe2?

Instructions on the separate pipelines may complete execution at the exact same time, thus the minimum time interval between completion can approach as little as 0 nanoseconds.

4. (5) The number of molecules in one mole of a substance is given by Avogadro's number = 6.022140857 * 10^23. The 64-bit floating point representation of this number on the MIPS processor is 0x44EFE185D2F54B66 without rounding. How is 6.022140857 * 10^23 represented in the 80-bit internal floating point format used by the IA-32 Intel processor? Express your answer as a 20-digit hex pattern.
Note that 6.022140857 * 10^23 = 602214085700000000000000 = 0x7f86174bd52d9b184000

Floor(log2(6.022140857 * 10^23)) = Floor(78.9946226353) = 78
6.022140857 * 10^23 / 2^78 = 1.9925592651895500159579
. 9925592651895500159579 * 2^63 = 9154743371470419504 (rounding to whole number)
= 111 1111 0000 1100 0010 1110 1001 0111 1010 1010 0101 1011 0011 0110 0011 0000 (this represents the fraction)
Characteristic = 78 + 16383 = 16461 = 100000001001101
Sign: 0
Integer part: 1
Full Binary: 0100 0000 0100 1101 1111 1111 0000 1100 0010 1110 1001 0111 1010 1010 0101 1011 0011 0110 0011 0000
Full Hex: 0x404DFF0C2E97AA5B3630

5. (4) A 16-core multi-core system is observed to provide superlinear speedup in executing a large application program when compared to executing the same program on a single-core system. Which one of the following values is the minimum speedup that the multi-core system might provide?
a) 14.4
b) 15.3
c) 17.1
d) 18.2

Observing a speedup greater than liner speedup is referred to as "superlinear" speedup, thus any speedup greater than 16 on a 16-core system would be considered superlinear. Thus, possible solutions include C and D. C has a lower speedup than D, thus is the minimum on the selected set. The solution is C.

6. Instruction windows as well as register windows have been described in this course. Which one of the following items (a through e) is facilitated most by the use of register windows and which one is facilitated most by the use of an instruction window? Use one of the labels I through V for each of your answers to a) and b) below.

I. register renaming

II. superscalar operation
III. procedure calls
IV. cache hits
V. data forwarding

a) (4) Instruction windows: II b) (4) Register windows: III

7. (4) Consider the following items:

a) GUI (graphical user interface)
b) Sticky Bit
c) Dirty Bit
d) MESI Bits

For each of the units or systems listed below, which one of the above items is most likely to be used by or for that unit or system? Use one of the labels a), b), c), or d) to indicate one selection for each of I through IV below.

I.	the floating point processor	B
II.	cache controller	D
III.	virtual memory system	C
IV.	input/output system	A

8. As you know, each set within a 4-way set associative cache contains 4 lines: Way0, Way1, Way2 and Way3. The order in which the lines within a full set were referenced (i.e., read, written or replaced) determines which line was most recently used down to least recently used. For example the reference order: Way2, Way3, Way0, Way1 corresponds to Way2 as the most recently used and Way1 as the least recently used line.

a) (2) What is the number of different reference orders for which Way1 is the least recently used?

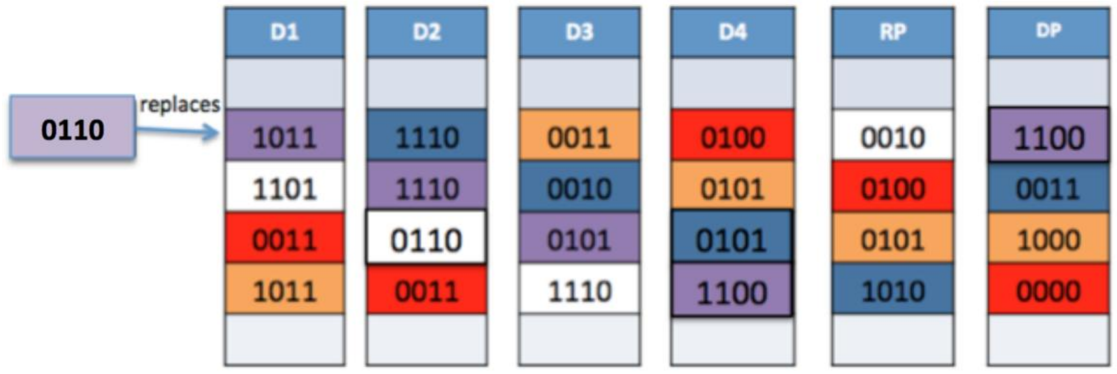
Way0, Way2, Way3, Way1
Way0, Way3, Way2, Way1
Way2, Way0, Way3, Way1
Way2, Way3, Way0, Way1
Way3, Way0, Way2, Way1
Way3, Way2, Way0, Way1
6 different reference orders

b) (3) What is the minimum number of bits required to correctly identify all of the possible reference orders for the 4 lines within each set?

There are a total of 24 different possible reference orders (1*2*3*4) for a 4-way set associative cache. 24 reference orders can be mapped to 5 bits ($24 < (2^5=32)$).

9. Shown below is a RAID-DP (i.e., RAID6 with dual parity) disk system. D1, D2, D3 and D4 are the data disks. RP is the horizontal or row parity disk and DP is the diagonal parity disk.

For the purposes of this problem, each strip or block on a disk is a 4-bit pattern. The horizontal parity block for each row is the exclusive-OR of the four 4-bit data disk blocks within the row.
The coloring scheme indicates which data blocks comprise each diagonal. Each diagonal parity block on the DP disk is the XOR of all the data blocks that are the same color as the diagonal parity block. Blocks are identified by the combination of the disk name (D1, D2, D3, D4, RP or DP) on which the block resides together with the block color (blue, purple, white, red or orange). The purple block on D1 (D1_purple) is to be overwritten with the new 4-bit pattern shown in the diagram.



(8) Name each block, other than D1_purple, that must be updated and indicate the 4-bit pattern contained in each updated block.

RP_white : 0010 -> 1111
DP_purple : 1100 -> 0001

10. (5) Assume that a 32-bit device status register and its associated 32-bit data register are mapped to adjacent memory addresses. Whenever a new data input is available, the device places the new input into its data register and sets the LSB within its status register to 1 to signal the presence of the new data value. Each time that the CPU reads data from the device's data register, the LSB within the status register is automatically cleared back to 0. The device can optionally be configured to generate an interrupt each time that the LSB in the status register is set. The I/O latency is defined as the time interval from when a new data input appears in the device data register and the time when the CPU reads the data from the device data register. Is this latency shorter if the I/O is performed using polling or is the latency shorter if the I/O is performed using interrupts? Explain your answer.

Polling is the process of checking device status registers periodically. If there is any change to the device status register when the CPU checks, then the CPU takes the action to read the data. Interrupt driven I/O signals are sent to notify the CPU on I/O completion. Since the interrupt driven I/O forces the CPU to take an action directly after the I/O value is set, while polling may have a significant delay between setting the I/O value and polling the status register to see if there is a change, the latency is shorter if the I/O is performed using interrupts.

11. (4) What is the minimum number of CPU registers that must be saved and restored by a routine that services an exception on our MIPS processor?

For any MIPS instruction, both the status register (12) and cause register (13) must be updated. Status register 14 may not be overwritten if the exception was caused by an external interrupt. Thus, just two registers must be saved (12, 13) to serve the simplest exception on the MIPS processor

12. P1 and P2 are the two processors in a dual processor system. Each processor has one 524288-byte data cache and uses 32-bit memory addresses. P1's cache is organized as a 1-way set associative cache and P2's cache is organized as a 2-way set associative cache. The cache line size and memory block size for both systems is 256 bytes.
Both caches contain an exact copy of memory block 502833. All of the lines within P2's cache are valid. The tags for way0 and way1 within every set in P2's cache are decimal 3761 and decimal 491 respectively. Assume that P1 then writes the value 0xCAFE1234 to address 0x07AC3144. Answer each of the following questions:

1-way set associative is just direct mapped
P1:
Offset bits = $\log_2(256)$ = 8 bits (assuming byte addressable system)
# lines = 524288 / 256 = 2048 lines
Line bits = $\log_2(2048)$ = 11 bits
Tag bits = 32 – 11 – 8 = 13 bits

P2:
Offset bits = (256) = 8 bits (assuming byte addressable system)
# lines = 524288 / 256 / 2 = 1024 lines
Line bits = $\log_2(1024)$ = 10 bits
Tag bits = 32 – 10 – 8 = 14 bits

a) (4) What is the line number of any line within P1's cache that is affected by the write?

0x07AC3144 = 0000 0111 1010 1100 0011 0001 0100 0100
Line number = 10000110001 = 1073

b) (4) For any line affected in P1's cache, what is the MESI state of the line before and after P1 performs the write?

The line affected is 1073. Since before the write, both caches contain the same memory block, the initial MESI state for the line in P1's cache is S – shared. After the write, the value is not yet in P2's cache, however it has been written to P1's cache, thus the MESI state is E – exclusive.

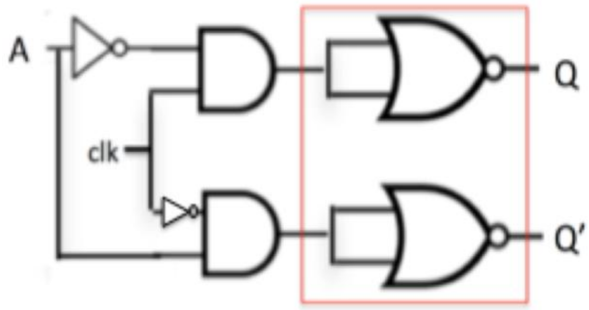
c) (4) What is the Set and Way for any line within P2's cache that is affected P1's write?

0x07AC3144 = 0000 0111 1010 1100 0011 0001 0100 0100
Line number = 0000110001 = 49
Tag = 00000111101011 = 491 -> cache hit
Way1 and line number 49 is affected by the write

d) (4) For any line affected in P2's cache, what is the MESI state of the line before and after P1 performs the write?

The line affected is 49. Since before the write, both caches contain the same memory block, the initial MESI state for the line in P2's cache is S – shared. After the write, the value is not yet in P2's cache, however it has been written to P1's cache, thus the MESI state is M - modified.

13. Consider the following circuit.



a) (2) Write down a logic expression that gives Q as a function of the input A when clk=1.

A = 0 -> Q = 0
A = 1 -> Q = 1
Q = A

b) (2) Write down a logic expression that gives Q' as a function of the input A when clk=0.

A = 0 -> Q' = 1
A = 1 -> Q' = 0
Q' = A'

c) (3) Is the circuit an example of a combinational circuit or an example of a sequential circuit? Explain your answer.

Combinational logic circuits generate outputs that depend only on the current set of inputs, while sequential logic circuits generate outputs that depend on the previous history of inputs. This logic circuit depends only on the inputs of A and clk, therefore is a combinational logic circuit.

14. (5) a) When an exception is caused by the execution of an instruction, that instruction is referred to as the offending instruction. Write down a MIPS instruction sequence that runs on our MIPS non-pipelined multi-cycle datapath and places the offending instruction into register $t0. Use as few instructions as possible.

```
mfc0 $t1, $14           # move the address of offending instruction to $t1
lw $t0, 0($t1)          # load the instruction at the given address
```

(3) b) Does the same instruction sequence work for our MIPS 5-stage pipelined system? Explain why or why not.

This instruction sequence will not work on our MIPS 5-stage pipelined system unless we add a hazard unit. The lw instruction depends on the result of the mfc0 instruction, thus will not produce the correct result unless we stall the lw instruction by two cycles. An instruction sequence that would work even without the hazard unit would be:

```
mfc0 $t1, $14           # move the address of offending instruction to $t1
nop                     # stall 1
nop                     # stall 2
lw $t0, 0($t1)          # load the instruction at the given address
```