ECE116C Final Fall 2019

Name:

UID:

Allotted Time: 2 hours 30 min. **Total Points: 70**

Instructions

1. **One sheet of paper (double sided, handwritten) allowed.**
2. **Calculators allowed.**
3. **MIPS reference sheet (detachable from front of your book) is also allowed.**
4. **-1point/min of late submission**
5. **Explain *all* your answers. Points will be deducted otherwise (even if the answer is correct). Cross out anything that you don't want read. Points will be deducted for incorrect scribbles.**

Q1. Memory Hierarchy 20 points (5 + 5 + 5 + 5)

You have written a "memory virus" program to stress test a processor's memory system. It only consists of bunch of LW instructions at different addresses. Assume that this processor has no virtual memory but has caches and the memory is byte addressable and the word size 32 bits.

(a) You notice that memory accesses either take 1 cycle or 10 cycles or 100 cycles. How many levels of cache does the memory system have ? What are the miss penalties for each level ?

L1, L2 and main memory.

L1 miss penalty = 9 cycles, L2 miss penalty = 90 cycles

(b) Second peculiar thing you observe is that with the following sequence of memory accesses (after restarting your program fresh): 0, 32, 4; you get access latencies as 100, 100, 1. What can you say about the cache organization from this observation ?

2-4 words per cacheline. Cannot be less than 2, cannot be more than 4.

(c) Next you do SW instruction on address 4 (after executing the LW instructions from (b)). You observe that the SW completes in 1 cycle. What further can you say about the cache organization ?

Write back L1 cache.

(d) Next you do a SW on address 80 followed by a LW on address 80. You observe that the LW takes 10 cycles. What more can you say about cache organization now ?

Write-allocate till L2, L1 is not write-allocate.

Q2 Storage and I/O 20 points (6 + 6 + 8)

(a) If the overheads of polling and interrupt handling are 100 and 1000 cycles respectively for a processor with clock frequency 100MHz, which is a better approach (in terms of CPU time wasted) for a mouse which is moved at most once every 10 milliseconds but on average moves only once every 10 seconds.

Polling: every 10 millisecond – 100 cycles
Interrupt: every 10 seconds – 1000 cycles

Polling - In 1 second – 1000/10*100 cycles = 10^4 cycles/second
Interrupt - In 1 second – 1/10*1000 cycles = 100 cycles/second (better approach)

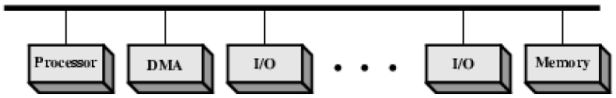
(b) To get to 2TB storage with 500GB disks, how many disks will I need in (i) RAID1 (ii) RAID5 configurations? If 1 disk fails, will I lose data in either case ? If two disks fail, will I lose data in either case ?

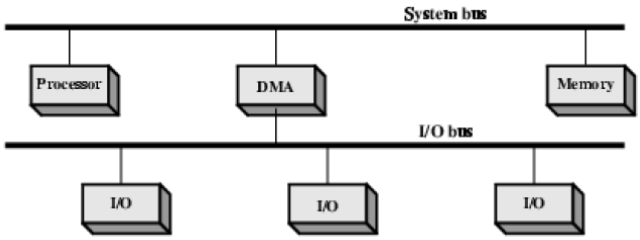
RAID1 – 4 disks + 4 mirrored disks
RAID5 – 4 disks + 1 parity disk

No, you will not lose data in either case if 1 disk fails

If two disks fail-
RAID1 – If the two failing disks are original and mirror of the same failing disk then you would lose data. For any other combination, you would not.
RAID5 – You would lose data.

(c) Consider two Memory-processor-DMA-IO bus organizations below.





(1) (2)

How many bus transactions would be needed in either case for an IO device to read/write to memory ? How many times would the CPU need to stall waiting for bus access to memory ?

In total there would be 4 bus transactions for both cases (I/O -> DMA -> Mem -> DMA -> I/O)
For the first case processor has to stall for all 4 transactions, in the second case the CPU would need to stall twice (DMA -> Mem and Mem -> DMA). The interactions between DMA and I/O use a separate bus.

(d) A 64 bit CPU (64 bit words and addresses) and memory share a 32-bit bus running at 100MHz. The memory needs 50ns to access a 64-bit value from one address. How many cycles does it take to read one 64bit word from memory ? If all memory accesses are random (i.e., no spatial locality), what is the effective memory bandwidth ? What is the bandwidth if the bus was widened to be a 64bit bus ? What is the bandwidth if all memory accesses are in bursts of 512 bits (for a 64 wide bus and contiguous addresses for the 512 bits) ?

To access a 64-bit value from the memory (32-bit bus):
2 cycles(send address) + 50ns(read data from memory) + 2 cycles(send the data) = 2+5+2 = 9 cycles (90 ns)
(a) Bandwidth (32-bit bus) = 64 bits / 90ns = 88.88 MB/sec

To access a 64-bit value from the memory (64-bit bus):
1 cycle(send address) + 50ns(read data from memory) + 1 cycle(send the data) = 1+5+1 = 7 cycles (70 ns)
(b) Bandwidth (64-bit bus) = 64bits/70ns = 114 MB/sec

(c) Soln1:
Burst access (512 bits) = 1+5+8 cycles (assuming accessing 512 contiguous bits from the memory is the same as accessing 64 bits – row buffer locality)
Bandwidth = 512bits/140ns = 457 MB/sec

Soln2:
Burst access = 1+5*(512/64) + 1 = 42 cycles (assuming that the previous 64-bits is sent while the next 64-bits is being accessed)
Bandwidth = 64B/420ns = 152 MB/sec

Soln3:
Burst access = 1+5*(512/64) + 8 = 49 cycles (assuming everything happens serially)
Bandwidth = 64B/490ns = 131 MB/sec

Q3. Parallelism 18 points (5 + 4 + 4 + 5)

(a) Consider a program that, when executing on one processor, spends 10% of its time in a non-parallelizable region. How much faster can this program run on a 3-processor system? Can this speedup be achieved in practice ? Why or why not ?

Single processor = T
3-processor system = 0.1T + 0.9T/3 = 0.4T
Speedup = T/0.4T = 2.5x

No – Synchronization overhead.

(b) Why does coarse-grained multi-threading offer in advantage over a single threaded processor ? What does fine-grained multi-threading buy us further ?

Coarse grained multi-threading over single threaded processor – better utilization of resources during long latency stalls (eg. cache miss).
Fine grained multi-threading – helps to remove data dependency stalls from the execution pipeline.

(c) Suppose we make our 5 stage pipeline fine-grain multithreaded with thread switching every cycle. How many threads does the processor need to keep in context to eliminate need for any forwarding hardware (including register file bypassing) ?

5 threads since jump on a register value will only be resolved after WB.

4 threads if jump on register is not considered.

(d) VLIW and Superscalar process both implement multiple-issue. Former being a compiler driven approach while latter relying on instruction scheduling in hardware. Suppose we have a dual pipeline which can execute two memory operations in parallel as long as they don't depend on each other. Consider the following program.

```
sw $7, 4($2)
lw $1, 8($5)
```

Can a VLIW machine issue the above two instructions together ? Why or why not ? Can a superscalar processor issue them together ?

VLIW – No because the final memory address to store into and load from in these two instructions might be the same, cannot resolve dependency.
Superscalar – Can resolve dependency.

Q4. Datapath & Arithmetic 12 points (5 + 2 + 5)

(a) In modern chips, it is very expensive to ensure that no errors ever happen. Imagine a MIPS processor where single precision floating point registers are especially error prone. Errors are modeled as flipping of exactly *one bit* in the 32-bit register. If we want the difference between the correct and incorrect values of the floating-point number to be less than 100, what is the minimum number of bits in the 32-bit register that should be protected from error? What are those bits?

All bits.

(b) Just like branch hazards, can a delay slot help reduce the overheads of exceptions ? Why or why not ?

No because you do not know which instruction will cause an exception.

(c) Where are no-ops needed to avoid stall in a classic 5-stage pipeline for the following piece of code. Assume no data forwarding except register file bypassing.

sub $2, $1,$3

and $4, $2,$5

or $8, $2,$6

add $9, $4,$2

slt $1, $6,$7

sub $2, $1,$3

Nop

Nop

and $4, $2,$5

or $8, $2,$6

Nop

add $9, $4,$2

slt $1, $6,$7