Computer Organization 2 Spring 2018

Final Study Guide

April 22, 2019

The test consists of

1. Multiple choice questions - 15 x 2 = 30 points
2. Pipeline simulation = 15 points
3. Cache Reference Table - 20 points
4. Fill out a branch prediction table - 10 points
5. Page Table Simulation - 10 points
6. Short answer questions - 5 x 6 = 30 points

- You will have an opportunity to earn 15 extra credit points.
- Please try and attempt all questions. You get points for trying.
- Anything from the slides/homework is fair game.
- Making me laugh might gain you points (depends on the quality of the joke).

Topics to study

- **Multi-Cycle Datapath and Control**
  - General steps performed by all instructions.
  - Datapath changes: single memory unit, single ALU unit, multiple temporary registers (IR, MDR, A, B, ALUout).
  - New control signals: IorD, ALUSrcA, ALUSrcB, IRWrite, PCWrite, PCWriteCond, PCSource
  - What are the actions taken on each of the following steps (be able to trace datapath for any instruction during any step and set control signals):
  - Instruction Fetch
  - Instruction Decode + Reg Fetch
  - Execution (for all instruction types)
  - Mem Access/ R-type completion
  - Read Completion
  - How many cycles does each instruction require?
  - Advantages and Disadvantages of multi-cycle. Be able to explain how it differs from single-cycle.

- **Pipelining**
  - Laundry analogy
  - Effect on throughput and instruction latency.
  - What is the ideal speedup? What factors prevent us from achieving ideal speedup?
  - What happens in each of the following stages? IF, ID, EX, MEM, WB
  - What are the datapath elements of a 5-stage pipeline? Note the addition of IF/ID, ID/EX, EX/MEM, and MEM/WB pipeline registers.
  - Be able to calculate speedup for a particular program vs single- or multi-cycle.
  - Define dependencies, hazards, and stalls.
  - Be able to explain and give an example of structural, data, and control hazards.
  - Solutions to data hazards: forwarding and stalls. When can forwarding be done without a stall? When is a stall required?
  - Solutions to control hazards: stalling and prediction methods.
  - Be able to identify dependencies and hazards within MIPS code snippets. Be able to identify what the solution should be to resolve the issue and practice reordering instructions to avoid hazards.
- **Pipelining Datapath and Control**
  - Know datapath elements, control signals, and pipeline registers for pipeline which does not handle hazards. Be able to take any instruction and chart its path through the pipeline across 5 cycles as well as indicate control signal values.
  - What information needs to be carried with the instruction? What are the fields of each pipeline register?
  - Know how control lines are grouped into EX, MEM, and WB and passed into pipeline registers.
  - If you understand all aspects of example walk-through at the end of the slides, you are good. Make sure you understand everything happening during each cycle!
- **Pipelining Hazards**
  - Data hazards: forwarding and stalling
  - Know the data hazard conditions listed on slide 12, as well as how to classify any data hazards into one of these 4 conditions. Does this make sense to you intuitively?
  - Know full data hazard conditions listed on slides 17 and 18 (revised on slide 29). Be able to explain in words each part of these equations and why/how these equations identify data hazards.
  - Forwarding unit: what are input and outputs? Understand conditions for values of ForwardA and ForwardB (outlined on slide 31). Do NOT memorize these you should be able to reconstruct this table with your understanding rather than memorization.
  - Know to stall immediately after load word with a dependent subsequent instruction (equation on 34).
  - How do we implement a stall?
  - Control hazards: be able to describe the problems in words.
  - Solutions: assume not taken, reduce delay by moving branch into ID stage, or prediction. Explain each approach. What are advantages/disadvantages of each approach?
  - How does dynamic branch prediction work?

- 1 and 2 bit branch prediction buffers how are they implemented? How do they work? Be able to explain behavior of code using these prediction buffers.
- Branch target buffer how does it work? How does it improve performance when used with branch prediction buffer?
- Be comfortable with final datapath and control. Given an instruction and a stage, be able to explain what is happening for that instruction in that stage as well as any hazard control activities that are happening.
- Project 2

- **Advanced ILP**
  - What are exceptions?
  - Difficulties with handling exceptions in pipeline: can occur out of order + multiple exceptions can occur in same clock cycle (be able to give examples).
  - What does it mean to support precise exceptions?
  - What are the basic steps to handle an exception in a pipeline? Know use of EPC and Cause registers.
  - Multiple cycle operations: why are many arithmetic operations (mult, FP ops, etc) not performed in one cycle?
  - Definitions: use latency and initiation interval. Be able to calculate these for instruction type.
  - Understand modified pipeline of slide 13 using pipelined mult and FP add, but unpipelined division.
  - Properties of multiple cycle operations (slide 15).
  - What is superpipelining? Multiple issue? Dynamic scheduling? Out of order execution processors?

- **Memory Hierarchy**
  - Desirable properties of memory: quick access and large size.
  - Temporal and Spatial Locality
  - General Memory Hierarchy Concepts
  - Four technologies used in memory.
  - Memory Hierarchy Terms
  - Caches
    - Direct-Mapped, Set Associative, Fully Associative
    - Offset, Index, Tag know how to calculate bit width and find values.
    - Relationships between cache attributes (e.g. block size and miss rate).
    - Block replacement strategies (random and LRU).
    - Write-through vs. Write-back.
    - Write allocation vs. No write allocate.
    - Cache miss categories: compulsory, capacity, and conflict.
    - Techniques for reducing miss penalties: critical word first and early restart.
    - Multi-level caches.
    - General techniques for improving cache performance (and their drawbacks).

- Virtual Memory
  - - What are the two motivations for virtual memory?
  - General Virtual Memory Concepts
  - Each process is compiled into its own virtual address space.
  - Virtual addresses are translated into physical addresses at run time.
  - Virtual Memory Terms: page, page fault.
  - Partitioning of virtual and physical address.
  - Process of translation from virtual to physical address.
  - Common design choices for virtual memory systems.
  - Page table concepts.
  - TLB concepts.
  - Be able to combine caching concepts to implement/interpret TLB.
  - Multiprogramming with virtual memory.

Some Sample Questions

1. How many offset bits are required for a cache with 32 word blocks?
   (a) 5
   (b) 6
   (c) 7
   (d) 8
2. Which of the following is the default start state for a branch on the branch predictor?
   (a) Strongly Not Taken
   (b) Weakly Not Taken
   (c) Strongly Taken
   (d) Weakly Taken
3. Look at the following MIPS code and fill out the Branch Prediction Table. Initially, populate them with the PC address of the branches, PC+4 in the branch target and WNT for the state. Then, show the changes to the table as we run through the code. At the end, specify the number of mispredicted branches.

```
        .text
        ori     $s0,$0,64
        ori     $t0,$0,0
        ori     $t1,$0,10
        ori     $t4,$0,0
HERE:   sll     $t2,$t0,2
        add     $t2,$s0,$t2
        lw      $t3,0($t2)
        beq     $t3,$0,DONE
        sll     $t3,$t3,3
        addi    $t0,$t0,1
        add     $t4,$t4,$t3
```

```
        sw      $t3,0($t2)
        bne     $t0,$t1,HERE
DONE:   li      $v0,1
        add     $a0,$0,$t2
        syscall
        .data
DATA0:  .word   20
DATA1:  .word   13
DATA2:  .word   -8
DATA3:  .word   0
```

PC	Branch Target	State

PC	Prediction	Action

4. Why do we need virtual memory management?

5. What is a fully associative cache? What are some of the advantages?

6. Consider the following instructions. Draw out the pipeline diagram, indicating the cycle in which each instructions stages are executed. Assume that forwarding is used to avoid stalls, when possible. Draw lines between stages to indicate forwarded values. If necessary, stalls may be indicated with an S or a shaded box. You only need to show one run through these lines of code. Assume the branch is NOT TAKEN at the bne. This is a loop, but you don't have to go on until the program halts. You can also assume the registers contain some valid data.

```
        .text
        ori     $s0,$0,36
        sw      $t1,0($s0)
        bne     $t5,$t6,LOOP
        add     $t2,$t0,$t1
        sw      $t2,$s0,17
        add     $t3,$t1,$t4
LOOP:   andi    $s1,$s0,15
        sll     $t0,$t1,5
        halt
        .data
Data1:  .word       12
Data2:  .word       8
Data3:  .word       17
```

7. Assume a 2-way set-associative cache with 16 cache sets, 1 word per block, and an LRU replacement policy. For both a write-through, no write-allocate and a write-back, write-allocate cache, fill in the appropriate information for the following memory references 92 (0101 1100)

28 (0001 1100)
94 (0101 1110)
24 (0001 1000)
88 (0101 1000)
216 (1101 1000)
26 (0001 1010)

R/W	Address	Tag	Index	Offset	Result	Memref?	Update?
R	92						
R	28						
W	94						
W	24						
R	88						
W	216						
R	26						
W	24						

Table 1: Write-Through, No Write-Allocate

R/W	Address	Tag	Index	Offset	Result	Memref?	Update?
R	92						
R	28						
W	94						
W	24						
R	88						
W	216						
R	26						
W	24						

Table 2: Write-Back, Write-Allocate

8. The following table is a stream of virtual addresses as seen on a system. Assume 1KB pages, an initially empty fully associative TLB with 4 lines and true LRU replacement. If pages must be brought in from disk, assign the next available physical page number (starting from 13).

9452	0010 0100 1110 1100
6388	0001 1000 1111 0100
10944	0010 1010 1100 0000
5540	0001 0101 1010 0100
6200	0001 1000 0011 1000
1244	0000 0100 1101 1100
5548	0001 0101 1010 1100

TLB:

Valid	Tag	Physical Page Number
0	11	12
1	7	1
1	3	5
0	4	4

Table 3: TLB

Virtual Page No	Valid	Physical Page Number or Disk
0	1	2
1	1	6
2	0	Disk
3	1	5
4	1	4
5	1	3
6	0	Disk
7	1	1
8	0	Disk
9	0	Disk
10	1	0
11	0	Disk

Table 4: Page Table

For each reference, indicate whether it was a hit in the TLB, a hit in the page table, or a page fault and what the physical page number is. If pages must be brought in from disk, assign the next largest physical page number (starting with physical page number 13). In other words, fill out the following table:

Virtual Addr	Virtual Page #	Page Offset	TLB Tag	TLB Result	Page Table Result	Physical Page #
9452						
6388						
10944						
5540						
6200						
1244						
5548						