UNT
UNIVERSITY OF NORTH TEXAS

CSCE 5610
Computer System Architecture

Exam Review

Outline

This is a review, not an in-depth coverage of every topic and concept we have gone over. Please review your notes, the slides and the assignments to make sure you are not missing anything for the exam.

Important

- Failure to follow these rules may result in you receiving an undesirable grade on your exam plus other consequences
- Calculator is needed.
- **No cell phones** are permitted for any reason!
- NO MAKEUP EXAM

Exam Format

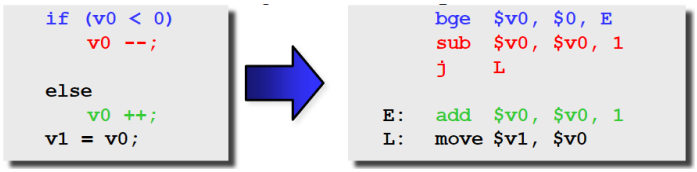
- The exam will cover all the content we discussed.
- Specifically, we will test the content from Slides02 – Slides09
- Exam I (November 7, 11:30am-12:30pm)
  - 5 True/False, multi-choice and multi-answer questions
  - 3 Calculation questions
  - 3-4 Concept questions
- Exam II (November 14, 11:30am-12:30pm)
  - 5 True/False, multi-choice and multi-answer questions
  - 3 Calculation questions
  - 3-4 Concept questions

Slide 02

- MIPS Assembly language
  - MIPS is a register-to-register, or load/store, architecture.
  - MIPS uses three-address instructions for data manipulation.
  - MIPS processors have 32 registers, each of which holds a 32-bit value.
  - MIPS memory is byte-addressable, which means that **each memory address references an 8-bit quantity**. Be careful, if A is an integer array, A[0] in address $s0, A[1] should be in address $s0+4.

Slide 02

- From high level code to MIPS Assembly language
  - Basic ALU operations
  - If-then-else:



```
if (v0 < 0)
    v0 --;

else
    v0 ++;
v1 = v0;
```

```
    bge  $v0, $0, E
    sub  $v0, $v0, 1
    j    L

E:  add  $v0, $v0, 1
L:  move $v1, $v0
```

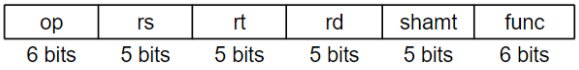
  - Loop:

```
for (i = 0; i < 4; i++) {
     // stuff
}
```

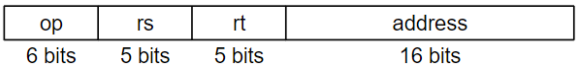
```
        add    $t0, $zero, $zero   # i is initialized to 0, $t0 = 0
Loop:   // stuff
        addi   $t0,$t0, 1          # i ++
        slti   $t1,$t0, 4          # $t1 = 1 if i < 4
        bne    $t1,$zero, Loop     # go to Loop if i < 4
```

Slide 03

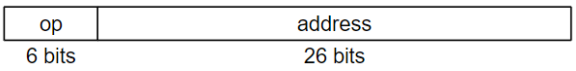
- Three instruction format
  - R-type: add, sub, etc.



  - I-type: addi, lw, swq, beq, etc.



  - J-type: j, jr, etc.



- Decode the Machine language
  - From Machine language to MIPS assembly code
- Single cycle MIPS processor
  - Trace the dataflow of the instructions.

Slide 04

- Single Cycle Performance
  - CPU Time.

$$\text{CPU time} = \text{CPU Clock Cycles} * \text{Clock Cycle Time} = \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

$$\text{CPU time}_{X,P} = \text{Instructions executed}_P * \text{CPI}_{X,P} * \text{Clock cycle time}_X$$

Cycles Per Instruction

  - Static instruction count vs. dynamic instruction count.

- Determine the clock cycle time based on the given execution time of each instruction

Slide 05

- Pipelining
  - Five pipeline stages.
  - Pipeline diagram
  - Pipelining performance

    Execution time on ideal pipeline:
    —time to fill the pipeline + one cycle per instruction

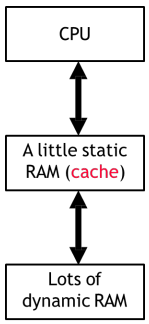
  - Insert NOP stages.
  - Pipeline performance
- Determine the clock cycle time based on the given execution time of each stage

Slide 06

- Hazards and pipeline bubbles
  - Structural hazards
    - result from not having enough hardware available to execute multiple instructions simultaneously
  - Data hazards (EXE/MEM, MEM/WB)
    - Hazards from R-type instructions can be avoided with forwarding.
    - Loads can result in a “true” hazard, which must stall the pipeline
  - Control hazards
    - arise when the CPU cannot determine which instruction to fetch next

Slide 07

- Cache
  - What is locality?
  - What is memory hierarchy, what are the levels of memory hierarchy?



  - What is cache? Types of cache? Direct mapped, set associative…

Slide 07

- Cache
  - What is cache? Types of cache? Direct mapped, set associative…
  - Performance metric: average access time
    - $AMAT = T_{hit} + \%miss * T_{miss}$
  - Memory and overall performance:

Memory stall cycles = Memory accesses x miss rate x miss penalty

CPU time = (CPU execution cycles + Memory stall cycles) x Cycle time

Slide 08-09

- Virtual Memory
  - Virtual Address
  - Physical Address
  - From virtual address to physical address