CIS 314

Final Review I

Shreeya Khadka, Brian Gravelle & Prof. Michel A. Kinsky
• Our fourth view of computer organization: The modern digital computer has three major functional hardware units: CPU, Main Memory and Input/Output (I/O) Units
Central Processing Unit (CPU) Organization

- CPU = Control Unit + ALU + Registers
- Control Unit: monitors and directs sequences of instructions
- ALU (Arithmetic-Logic Unit): performs arithmetic and logical operations
- Registers are memory cells built right into the CPU for temporary data needed by the ALU and for PC
- Clock synchronizes CPU operations
Central Processing Unit (CPU) executes program code in the following fashion:

1. Fetch Instruction
2. Decode Instruction
3. Execute Operation
4. Memory Operation
5. Register Writeback Operation
Instruction Fetch

- Program Counter (PC) sends current instruction address to memory to fetch instruction to execute

Diagram:

- PC (Program Counter) sends address to memory.
- Memory reads instruction [31-0] at the address.
- Instruction is fetched.
- PC is incremented by 4 for the next instruction.

32-bit Register

ADD

Instruction Memory

Instruction [31-0]
Instruction Decode

• Identity instruction class
  ‣ If R-Type, execution will need to:
    • Read two register operands
    • Perform arithmetic/logical operation
    • Write register result
  ‣ If Load/Store, execution will need to:
    • Read register operands
    • Calculate address using 16-bit offset
    • Load: Read memory and update register
    • Store: Write register value to memory
Instruction Decode

• Identity instruction class
  ▸ If Branch, execution will need to:
    • Read register operands
    • Compare operands
      ▸ Use ALU to subtract and check Zero output

• Calculate target address
  ▸ Sign-extend displacement
  ▸ Shift left 2 places (word displacement)
  ▸ Add to PC + 4
Execute Operation

- The Arithmetic Logic Unit (ALU) is at the center of the CPU operation execution

  - ALU operation is based on instruction type and function code
    - Performs subtraction for branches (beq)
    - Performs no operation for jumps
    - Performs the operation is specified by the function field for R-type instructions

  - ALU Control unit will have the following inputs:
    - 2-bit control field called ALUOp
    - 6-bit function field
Memory Operation

- For MIPS Load and Store are the only two memory instructions
  - Recall:
    - MIPS does not support memory to memory data processing operations
    - Data values must be moved into registers before using them
    - The basic load and store instructions are Load and Store Word or Byte
      - lw/lb  $rt, offset($rs)
      - sw/sb  $rt, offset($rs)
Register Writeback

- Write the resulting data from the instruction execution back to the register file
  - R-Type instructions
  - Load instructions
  - Some jump instruction (jal/jalr)
CPU Performance

• CPU performance factors
  ‣ Instruction count
  ‣ Determined by ISA and compiler
  ‣ CPI and Cycle time
  ‣ Determined by CPU hardware
  ‣ Longest delay determines clock period
    • Critical path: load instruction
CPU Performance

• Longest delay determines clock period
  ▸ Critical path: load instruction
    1. Instruction memory
    2. Register file read
    3. ALU operation
    4. Data memory access
    5. Register file writeback

• Performance can be improved by pipelining
Multi-Cycle MIPS Processor

• Single cycle processor
  ‣ The cycle time has to be long enough for the slowest instruction to complete
  ‣ CPI = 1

• Multi-cycle Implementation
  ‣ Instructions are broken into smaller steps
  ‣ Execute each step (instead of the entire instruction) in one cycle
  ‣ The cycle time has to be long enough for the slowest step to complete
Multi-Cycle MIPS Processor

• The advantages of the multiple cycle processor
  ‣ Cycle time is much shorter
  ‣ Different instructions take different number of cycles to complete
    • Load takes five cycles
    • Jump only takes three cycles
    • CPI is between 3 and 5

\[
\text{j label} \\
\text{lw rt, index(rs)}
\]
Multi-Cycle MIPS Processor

- Although CPI increased overall execution time is shortened, so performance is improved!

\[
\text{Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Time}}{\text{Cycle}}
\]

- Performance can be improved even more with pipelining!
Instruction Interactions

• An instruction in the pipeline may need a resource being used by another instruction in the pipeline
  ‣ Structural hazard

• An instruction may depend on something produced by an earlier instruction
  ‣ Dependence may be for a data calculation
    • Data hazard
  ‣ Dependence may be for calculating the next address
    • Control hazard (branches, interrupts)
Resolving Data Hazards

• Strategy 1: Wait for the result to be available by freezing earlier pipeline stages
  ‣ interlocks

• Strategy 2: Route data as soon as possible after it is calculated to the earlier pipeline stage
  ‣ bypass
• Speed Up Equations for Pipelining

\[
\text{CPI}_{\text{pipelined}} = \text{Ideal CPI} + \text{Average Stall cycle per Instruction}
\]

\[
\text{Speedup} = \frac{\text{Pipeline Depth}}{1 + \text{Pipeline stall CPI}} \times \frac{\text{Clock Cycle}_{\text{Unpipelined}}}{\text{Clock Cycle}_{\text{Pipelined}}}
\]
Memory Technology

- Random Access Memory (RAM)
  - Any byte of memory can be accessed without touching the preceding bytes
  - RAM is the most common type of memory found in computers and other digital devices
  - There are two main types of RAM
    - DRAM (Dynamic Random Access Memory)
      - Needs to be “refreshed” regularly (~ every 8 ms)
      - 1% to 2% of the active cycles of the DRAM
      - Used for Main Memory
    - SRAM (Static Random Access Memory)
Memory Technology

• Random Access Memory (RAM)
  ‣ Any byte of memory can be accessed without touching the preceding bytes
  ‣ RAM is the most common type of memory found in computers and other digital devices
  ‣ There are two main types of RAM
    • DRAM (Dynamic Random Access Memory)
    • SRAM (Static Random Access Memory)
      ‣ Content will last until power turned off
      ‣ Low density (6 transistor cells), high power, expensive, fast
      ‣ Used for caches
Memory Organization

- A memory cannot be large and fast
- Increasing sizes of cache at each level

- A hit at a level occurs if that level of the memory contains the data needed by the CPU
- A miss occurs if the level does not contain the requested data
Multilevel Caches

For a cache with hit rate $h$, effective access time is:

$$C_{\text{eff}} = hC_{\text{fast}} + (1 - h)(C_{\text{slow}} + C_{\text{fast}}) = C_{\text{fast}} + (1 - h)C_{\text{slow}}$$
Caches

- Local miss rate = misses in cache / accesses to cache
- Global miss rate = misses in cache / CPU memory accesses
- Misses per instruction = misses in cache / number of instructions
Memory Impact on Performance

• Suppose a processor executes at ideal CPI = 1.1 50% arith/logic, 30% load/store, 20% control and that 10% of data memory operations miss with a 50 cycle miss penalty

\[ CPI = \text{ideal CPI} + \text{average stalls per instruction} \]

\[ = 1.1 \text{(cycle)} + (0.30 \text{ (data memory operation / total instruction)} \times 0.10 \text{ (miss/data memory operation)} \times 50 \text{ (cycle/miss)} \) \]

\[ = 1.1 \text{ cycle} + 1.5 \text{ cycle} = 2.6 \]

• So 58% of the time the processor is stalled waiting for memory!
• This organization works because most programs exhibit locality

› The principle of temporal locality says that if a program accesses one memory address, there is a good chance that it will access the same address in the near future

› The principle of spatial locality says that if a program accesses one memory address, there is a good chance that it will also access other nearby addresses
Caching principles

• Compulsory misses
  ‣ With on-demand fetching, first access to any item is a miss

• Capacity misses
  ‣ We have to evict some items to make room for others
  ‣ This leads to misses that are not incurred with an infinitely large cache

• Conflict misses
  ‣ The placement scheme may force us to displace useful items to bring in other items
  ‣ This may lead to misses in future
**Placement Policy**

- **Cache**: Fully Associative anywhere
- **Set Number**: (2-way) Set Associative anywhere in set 0
- **Block Number**: Direct Mapped only into block 4

Block 12 can be placed in set 0 anywhere in memory (12 mod 4) and only into block 4 (12 mod 8).