CPS104
Review of Important Course Topics

Robert Wagner
Lecture 2: C and C++: Computer Memory

- Memory is a large linear array of 8 bit bytes.
  * Each byte has a unique address (location)
- Bytes are grouped into longer sequences, some of which can be addressed as one unit. Their addresses must be multiples of their byte lengths.
  * Byte (1 byte) -- characters (char)
  * Word (4 bytes) -- integers (int), floats.
  * Double (8 bytes)
Lecture 3: Data representations

- Introduction: Decimal, Binary, Octal and Hexadecimal numbers.
- Storage types: Byte, Word, Double-word.
- ASCII Characters.
- 2’s complement numbers.
- Floating-point numbers.
- Computer Instructions.
- Memory Addresses.

Sections 3, 7, 4.1-4.3, 4.8 In the textbook
Lecture 6: MIPS ISA and Assembler:

- The MIPS Assembly Language.
- MIPS Assembly Language Programming Conventions.
- The program Stack
- Useful C techniques: “case” selection, “hash lookup”

★ Reading Assignment: Chapter 3, Appendix A
★ SPIM manual.
Problem P6:
MYMIPS: Instruction Set Architecture, Simulator:
Lecture 12: A MYMIPS CPU Data Path:
MYMIPS ISA Subset

<table>
<thead>
<tr>
<th>OP 31..28</th>
<th>Name</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>Load</td>
<td>R[D]=MEM[R[A]+E()]; PC = PC+4</td>
</tr>
<tr>
<td>0011</td>
<td>Store</td>
<td>MEM[R[A]+E()]=R[D]; PC = PC+4</td>
</tr>
<tr>
<td>0101</td>
<td>Add</td>
<td>R[D]=R[A] + E(); PC = PC+4</td>
</tr>
<tr>
<td>0111</td>
<td>And</td>
<td>R[D]=R[A] &amp; EU(); PC = PC+4</td>
</tr>
<tr>
<td>1110</td>
<td>BrCond</td>
<td>PC = (T(D,R[A])) ? E() : PC+4</td>
</tr>
</tbody>
</table>

D=(29:24), A=(23:20), I=(19), IMM=(18:0), B=(3:0)
E() = I ? SignExt(IMM) : R[B]; EU() = I ? IMM : R[B]

<table>
<thead>
<tr>
<th>D:</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>T(D,X):</td>
<td>1</td>
<td>X&lt;0</td>
<td>X==0</td>
<td>X&lt;=0</td>
<td>X&gt;0</td>
<td>X!=0</td>
<td>X&gt;=0</td>
<td>0</td>
</tr>
</tbody>
</table>
Lecture 9: Integer Arithmetic: **Multiplication Algorithm #2**

1. Test Multiplier0
   - Multiplier0 = 1
     - 1a. Add multiplicand to the left half of the product and place the result in the left half of the Product register
   - Multiplier0 = 0

2. Shift the Product register right 1 bit

3. Shift the Multiplier register right 1 bit

32nd repetition?
   - No: < 32 repetitions
   - Yes: 32 repetitions

Done

Figure Copyright Morgan Kaufmann
Lecture 10: Boolean Algebra & gates:

- Truth tables, Boolean functions, Gates and Circuits
- Karnaugh maps for simplifying Boolean equations
- Examples: 2-1 MUX, Full Adder

Read Appendix B
Lecture 11: Gates, Buses, Latches:

- The MIPS ALU
- Shifter
- The Tristate driver
- Bus Interconnections
- Register Cell
- The Register File

Read Appendix B
Lecture 12: A MYMIPS CPU Data Path:

- **Designing a CPU**
  - From description of ISA (MYMIPS subset) to Hardware “Program”
  - Note patterns common to many instructions
    → Develop signals to indicate when these occur
  - Specify major CPU components
    → ALU, Register file, Memory must be tailored to MYMIPS
  - Overall CPU operation
  - Specify circuits using shorthand notation (RTL)
  - Translate RTL to circuits
    → BX signal computation
    → MYMIPS ALU, incl bi-directional shifter
    → Tailored Register File
  - Examine overall design for flaws

Read Appendix B
RFB is a 32-bit Latch, set on CLK from ALU or C (Memory) via MUX
NCLK is True when CLK is False, and Vice V
NCLK and CLK are never True at once

RFB Drives C during NCLK
All other drivers of C are OFF during NCLK
Lecture 13: The Memory System:
Levels of the Memory Hierarchy

<table>
<thead>
<tr>
<th></th>
<th>Capacity</th>
<th>Access Time</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Registers</td>
<td>100s Bytes</td>
<td>&lt;10s ns</td>
<td>~$200/MByte</td>
</tr>
<tr>
<td>Cache</td>
<td>K Bytes</td>
<td>10-100 ns</td>
<td>~$200/MByte</td>
</tr>
<tr>
<td>Main Memory</td>
<td>M Bytes</td>
<td>100ns-1000ns</td>
<td>~$6/Mbyte</td>
</tr>
<tr>
<td>Disk</td>
<td>G Bytes</td>
<td>10,000,000ns</td>
<td>~$100/GByte</td>
</tr>
<tr>
<td>Tape</td>
<td>infinite</td>
<td>sec-min</td>
<td>~$5/GByte</td>
</tr>
</tbody>
</table>

- Faster: Staging Xfer Unit
- Larger: Prog./compiler, Cache Ctrl, OS, User/Operator Mbytes

Diagram:
- Registers
- Instr. Operands
- Cache
- Blocks
- Memory
- Pages
- Disk
- Files
- Tape
Lecture 14: The Cache

- The Memory Hierarchy
- Direct Mapped Cache.
- Two-Way Set Associative Cache
- Fully Associative cache
- Replacement Policies
- Write Strategies
Lecture 16: Virtual Memory

Virtual Memory.
- Paged virtual memory.
- Virtual to Physical translation: The page table.
- Fragmentation.
- Page replacement policies.
- Reducing the Virtual to Physical address translation time.
  - The TLB
  - Parallel access to the TLB and Cache
- Memory Protection
- Putting it all together: The SPARC-20 memory system
Lecture 17: Interrupts, Exceptions and Traps

- Interrupts, Exceptions and Traps are asynchronous changes in the control flow. Interrupts and Exceptions can be viewed as asynchronous (unscheduled) procedure calls.

- Interrupts and exceptions are designed to provide:
  - protection mechanisms: Error handling, TLB management.
  - efficiency: Overlap I/O and execution, . . .
  - Illusion of parallelism: Timesharing, multithreading
  - Ability to handle Asynchronous external events: Network connections, keyboard input, DMA I/O, . . .
Lecture 17: Interrupts, Exceptions and Traps

- **Exception**: a change in execution caused by a condition that occurs within the processor.
  - segmentation fault (access outside program boundaries, illegal access, . . .)
  - bus error
  - divide by 0
  - overflow
  - page fault (virtual memory…)

- **Interrupt**: a change in execution caused by an external event
  - devices: disk, network, keyboard, etc.
  - clock for timesharing (multitasking)
  - These are useful events, must do something when they occur.

- **Trap**: a user-requested exception
  - Operating system call (syscall)
  - Breakpoints (debugging mode)
Lecture 18: Input-Output

- The I/O system
- Magnetic Disk
- Magnetic Tape
- Buses
- Direct Memory Access
Lecture 19: Pipelining

The Five Stages of Load

- **Ifetch**: Instruction Fetch
  - Fetch the instruction from the Instruction Memory
- **Reg/Dec**: Register Fetch and Instruction Decode
- **Exec**: Calculate the memory address
- **Mem**: Read the data from the Data Memory
- **WrB**: Write the data back to the register file
The five independent functional units in the pipeline datapath are:

- Instruction Memory for the Ifetch stage
- Register File’s Read ports (bus A and bus B) for the Reg/Dec stage
- ALU for the Exec stage
- Data Memory for the Mem stage
- Register File’s Write port (bus W) for the WrB stage

One instruction enters the pipeline every cycle

- One instruction comes out of the pipeline (completed) every cycle
- The “Effective” Cycles per Instruction (CPI) is 1; ~1/5 cycle time