Lecture Topics

- Today: The MIPS ISA
  (P&H 2.1-2.14)

- Next: Single-Cycle Processors
  (P&H 4.1-4.4)

Announcements

- Milestone #2 (due 2/2)
- Milestone #3 (due 2/9)
- Milestone #4 (due 2/23)
MIPS Instructions

- All instructions are 32 bits wide
- Three general categories:
  - Data manipulation (ALU operations)
  - Data movement (Load and Store operations)
  - Control transfer (branches and jumps)
- Three instruction formats (instructions from three categories spread across the three formats)
Data Movement

- Load/Store architecture – only specific instructions (loads and stores) access memory
- Load: copy byte(s) from memory to register
- Store: copy byte(s) from register to memory

Data Movement

- Loads
  - LW load word (4 bytes)
  - LH load signed halfword (2 bytes)
  - LHU load unsigned halfword (2 bytes)
  - LB load signed byte (1 byte)
  - LBU load unsigned byte (1 byte)
- Signed loads use sign extension
- Unsigned loads use zero extension
Data Movement

- Stores
  - SW  store word (4 bytes)
  - SH  store halfword (2 bytes)
  - SB  store byte (1 byte)

- Store instructions do not have to deal with sign or zero extension – not relevant

Data Movement

- Target address is always the sum of the contents of a register and the 16-bit immediate value (sign extended)

- Load:
  \[ R[rt] \leftarrow= \text{Mem}[ R[rs]+\text{imm16} ] \]

- Store:
  \[ R[rt] \rightarrow= \text{Mem}[ R[rs]+\text{imm16} ] \]
Control Transfer Instructions

- Allows the programmer to alter the normal flow of execution (sequential)

- Control structures:
  - selective control (if statements)
  - repetitive control (loops)

- Subroutines (functions):
  - call subroutine
  - return from subroutine

MIPS Jump Register Instructions

R-format

<table>
<thead>
<tr>
<th>Format</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JR</td>
<td>R[rs] ==&gt; PC</td>
<td>Jump to current instruction address</td>
</tr>
<tr>
<td>JALR</td>
<td>PC+8 ==&gt; R[rd]</td>
<td>Jump to subroutine address</td>
</tr>
<tr>
<td></td>
<td>R[rs] ==&gt; PC</td>
<td>Jump and save return address</td>
</tr>
</tbody>
</table>
MIPS Jump Register Instructions

- JR and JALR unconditionally jump to the address in R[rs]
- There is a *one instruction delay* – the transfer of control occurs after the following instruction is executed
- JALR saves the return address in R[rd]; the return address is PC+8 due to the delay slot

Example: JALR and JR

```assembly
# Address of "whatever" in $8
jalr $31, $8
nop
add $10, $11, $12
.
whatever:
.
jr $31
nop
```
MIPS Jump Instructions

J-format

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>000010</td>
<td>imm26</td>
<td>jump</td>
</tr>
<tr>
<td>3</td>
<td>000011</td>
<td>imm26</td>
<td>jal jump and link</td>
</tr>
</tbody>
</table>

- **J**: target address ==> PC
- **JAL**: PC+8 ==> R[31]
- target address ==> PC

MIPS Jump Instructions

- J and JAL unconditionally jump to the target address
- There is a one instruction delay
- JAL saves the return address in R[31]
- The target address is computed as:
  
  \[ \text{PC}[31:28] \ || \text{imm26} \ || \ 00 \]
Jump Target Address

- The target address is computed as:

\[
\text{PC}[31:28] \ || \ \text{imm26} \ || \ 00
\]

- Permits jump to any address in the same 256 MB area of memory as the PC

Example: JAL and JR example

```
jal whatever
nop
add $10, $11, $12
.
whatever:
  
  jr $31
  nop
```
MIPS Branch Instructions

I-format

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01</td>
<td>000001</td>
<td>rs</td>
<td>00000</td>
<td>imm16</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>000001</td>
<td>rs</td>
<td>00001</td>
<td>imm16</td>
</tr>
</tbody>
</table>
| 1 | 01 | 000001 | rs | 10000 | imm16 | lb | branch on rs < 0
| 1 | 01 | 000001 | rs | 10001 | imm16 | lbne |
| 4 | 04 | 000100 | rs | rt | imm16 | beq |
| 5 | 05 | 000101 | rs | rt | imm16 | bne |
| 6 | 06 | 000110 | rs | 00000 | imm16 | bgtz |
| 7 | 07 | 000111 | rs | 00010 | imm16 | bogt | branch on rs > 0

- Branches conditionally jump to the target address
  - true: PC + offset ==> PC
  - false: fall through
- There is a one instruction delay
- "AL" saves the return address in R[31]
Branch Target Address

- The target address is computed as:
  \[ PC + (IR[15:0] \| 00) \Rightarrow PC \]
- Relative to the PC
- Offset is "imm16" from IR (sign extended)
- Displaced by 2 bits (all word addresses end in 2 bits of zero)

Example: Branch example

```
loop:
    add $14, $14, $13
    addi $15, $15, -1
    bgtz $15, loop
    nop
```
MIPS Lite Considerations

- No operating system or standard libraries
- Memory layout:
  - 0x00000000 – text (object code)
  - 0x0000E000 – data
- Memory-mapped I/O
  - 0x0000F000 – output ASCII character
  - 0x0000F001 – output integer

MIPS Assembly Language

- Similar to most assembly languages
  - Symbolic machine instructions
  - Pseudo instructions
  - Assembler directives

- Examples on course website:
  - ~cse420/Examples/example04
  - ~cse420/Examples/example05