CMPE011 Fall 2002
MIPS Addressing Modes

- Memory organization
- MIPS addressing modes
- PowerPC additional addressing modes

Textbook: 3.8
Memory organization and addressing

- memory is viewed as a single-dimensional array of bytes individually addressable — 32-bit words are \textit{aligned} to 4-byte boundaries (instructions)

\begin{itemize}
  \item $2^{32}$ bytes, with addresses from 0 to $2^{32} - 1$
  \item $2^{30}$ words, with addresses 0, 4, 8, \ldots, $2^{32} - 4$
\end{itemize}
Memory organization

Byte ordering within words

- **Little-endian**: word address is LSB
- **Big-endian**: word address is MSB
MIPS addressing modes

MIPS addressing modes

Adressing modes are the ways of specifying a location of an operand, a location in memory, or the address of an instruction for a control transfer.

- register addressing
- immediate addressing
- base addressing
- PC-relative addressing
- indirect addressing
- pseudodirect addressing
**Register addressing**: specify the register number

Example: `add $t0, $t1, $t2`

```
<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>shamt</th>
<th>funct</th>
</tr>
</thead>
</table>
```

5 bits 5 bits 5 bits

5 bits $\leftrightarrow$ 32 registers
Register addressing:
**Immediate Addressing:** the operand is embedded inside the encoded instruction

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>immediate value</th>
</tr>
</thead>
</table>

16-bit signed integers range from \( -2^{15} \) to \( 2^{15} - 1 \)
16-bit unsigned integers range from \( 0 \) to \( 2^{16} - 1 \)

Example: `addi $t0, $t1, 77`
Immediate addressing:
**MIPS addressing modes**

**Base (or displacement) addressing:** the address is the sum of the immediate (16 bits, two’s complement) and the value in a register \((rs)\)

\[
\text{ADDR} = \text{byte offset} + (rs)
\]

```
  op  rs  rt  byte offset
```

16 bits

<table>
<thead>
<tr>
<th>lower mem.</th>
<th></th>
<th>higher mem.</th>
</tr>
</thead>
</table>

\((rs) - 32768\)  \((rs)\)  \((rs) + 32767\)

Example: \(\text{lw} \quad \$s0, 12(\$s1)\)
Base addressing:
MIPS addressing modes

PC-relative addressing: the value in the immediate field
(a two’s complement 16-bit number) is interpreted as an offset
in number of instructions with respect to the address
of the next instruction (PC + 4).

\[
\text{ADDR} = \text{offset} \ll 2 + \text{PC} + 4
\]

\[\begin{array}{ccc}
\text{op} & \text{rs} & \text{rt} & \text{instr. offset} \\
\hline
\text{lower mem.} & & \text{higher mem.} & \\
\end{array}\]

16 bits

\[\begin{array}{c}
\text{PC-131072} & \text{PC+4} & \text{PC+131068} \\
\end{array}\]
MIPS addressing modes

Example:

```
...  
addi $s0, $s0, 1
beq $s0, $s1, label
addi $s0, $s0, 1
addi $s0, $s0, 1
label: addi $s0, $s0, 1
addi $s0, $s0, 1
...  
```

The binary coding of the line `beq $s0, $s1, label` is actually `0x12110002`, i.e. 2 instructions after the next one (PC+4).

NOTE on SPIM: SPIM uses the current PC and not the next
PC-relative addressing:
**MIPS addressing modes**

**Indirect addressing:** the address is the value in a register.

Also called "register direct".

In MIPS, it is used in the *jump register* instructions \((jr, jalr)\)

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rd</th>
<th>0</th>
<th>funct</th>
</tr>
</thead>
</table>

Example: \(jr \ \$ra\)

AND it is a special case of base addressing (offset = 0)

Example: \(lw \ \$s0, 0($s1)\)
Indirect addressing:

![Diagram showing indirect addressing]

- **PC**
- **CPU REG**
- **Instruction**
- **MEMORY**
**MIPS addressing modes**

**Direct addressing:** the address is the immediate (no in MIPS)

**Pseudodirect addressing:** the 26-bit immediate field of the jump instruction (j, jal) is the instruction offset within the 256 MB page given by the four most significant bits of the PC.

```
   op  instruction offset
       26 bits

   256 MB page

Example:

<table>
<thead>
<tr>
<th>PC:</th>
<th>0110 0010010...</th>
</tr>
</thead>
<tbody>
<tr>
<td>offs:</td>
<td>00110011001100110011001100110010001000</td>
</tr>
<tr>
<td>&lt;shift&gt;:</td>
<td>00</td>
</tr>
<tr>
<td>ADDR:</td>
<td>0110 0011001100110011001100110011001100 00</td>
</tr>
</tbody>
</table>
```
Pseudo-direct addressing:
MIPS addressing modes

NOTE THAT:
**Addressing mode ≠ instruction type (format)**

Example:

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>TYPE</th>
<th>ADDRESSING MODE(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>addi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>beq</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PowerPC additional addressing modes

Indexed addressing: the address is the sum of two registers

Example (p. 175):

The MIPS code:

```
add $t0, $a0, $s3  # $a0 has base of an array, $s3 is an index
lw  $t1, 0($t0)  # $t1 gets ($a0+$s3)
```

can be replaced by the single PowerPC instruction

```
lw  $t1, $a0+$s3  # $t1 gets ($a0+$s3)
```

• pointer offset from a pointer
• does not require additional hardware
Indexed addressing:
PowerPC additional addressing modes

Update addressing: base addressing with automatic base register increment.

Example (p. 176):

The MIPS code:

```
lw   $t0, 4($s3)  # $t0 gets $s3+4)
addi $s3, $s3, 4  # $s3 points to the next word
```
can be replaced by the single PowerPC instruction

```
lwu  $t0, 4($s3)  # $t0 gets $s3+4), $s3 = $s3+4
```

- does not require new hardware, but
- does require to write two registers at the same time
Update addressing:
One more addressing mode

One more addressing mode (not in PowerPC)

**Memory indirect addressing:** a register points to a memory location that points to the operand.

Example, PDP-11 assembly language: \texttt{move $8, @$($3)\)}
Recommended exercises

- Ex. 3.12 to 3.17, 3.19, 3.20