

---

---

# CMPE011 Fall 2003

## MIPS Addressing Modes

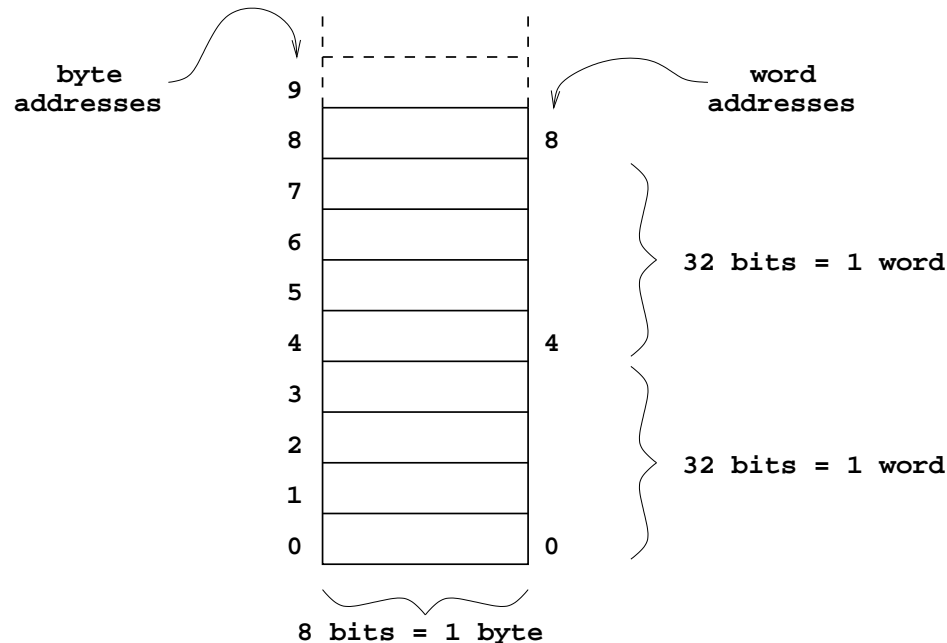
- Memory organization
- MIPS addressing modes
- PowerPC additional addressing modes

Textbook: 3.8



## Memory organization and addressing

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

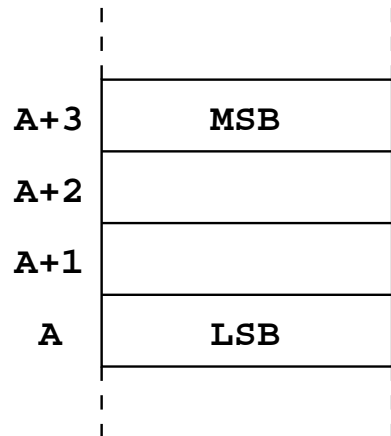
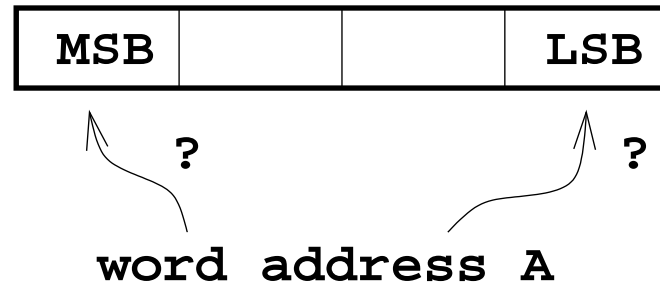


- $2^{32}$  bytes, with addresses from 0 to  $2^{32} - 1$
- $2^{30}$  words, with addresses 0, 4, 8, ...,  $2^{32} - 4$

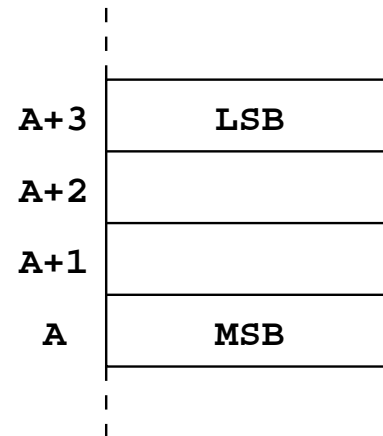


## Byte ordering within words

- *Little-endian*: word address is LSB
- *Big-endian*: word address is MSB



little-endian



big-endian



## **MIPS addressing modes**

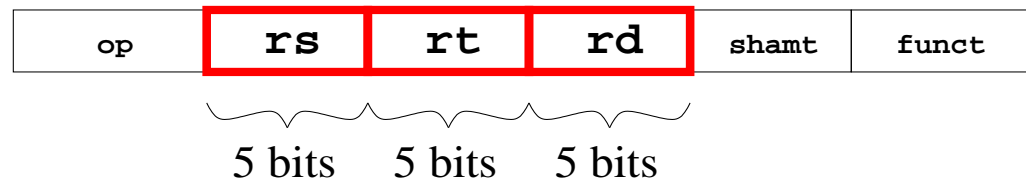
Addressing modes are the ways of specifying a location of an operand, a location in memory, or the address of an instruction for a control transfer.

- register addressing
- immediate addressing
- base addressing
- PC-relative addressing
- indirect addressing
- pseudodirect addressing



**Register addressing:** specify the register number

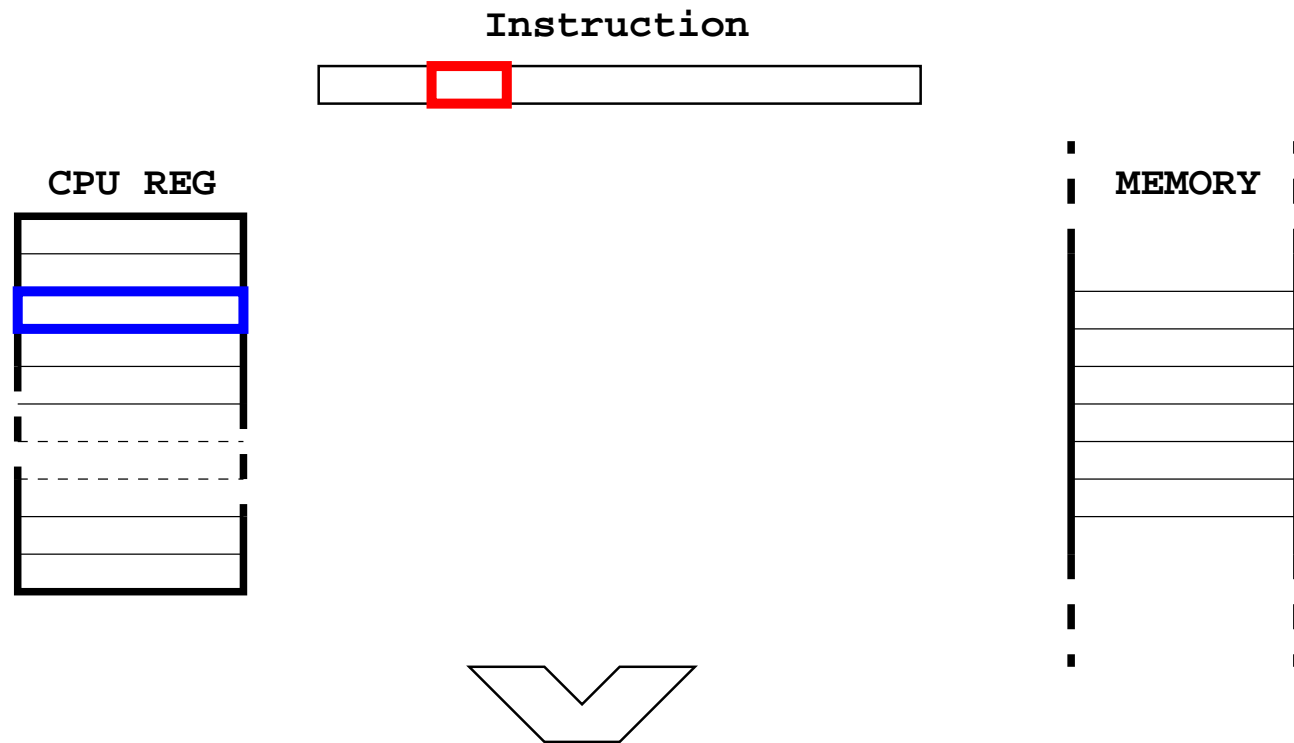
Example: add      \$t0, \$t1, \$t2



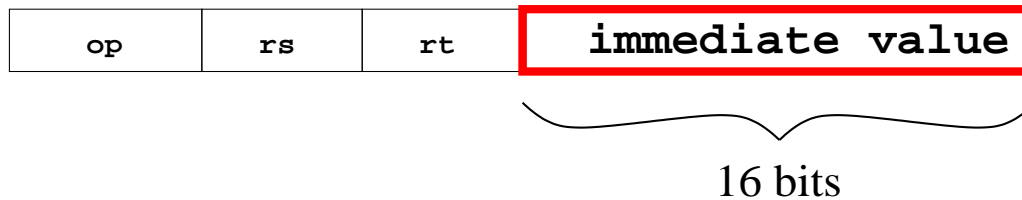
5 bits  $\iff$  32 registers



## Register addressing:



**Immediate Addressing:** the operand is embedded inside the encoded instruction



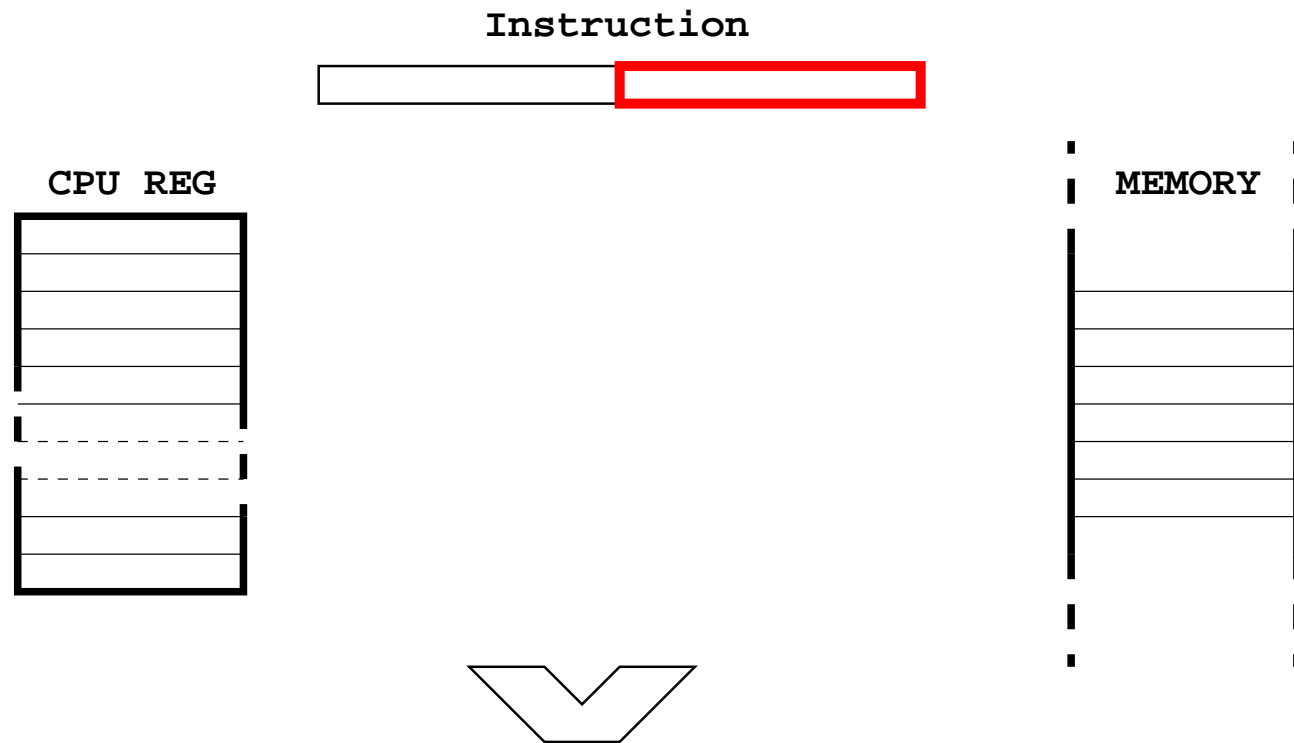
16-bit signed integers range from \_\_\_\_\_ to \_\_\_\_\_

16-bit unsigned integers range from \_\_\_\_\_ to \_\_\_\_\_

Example: `addi $t0, $t1, 77`

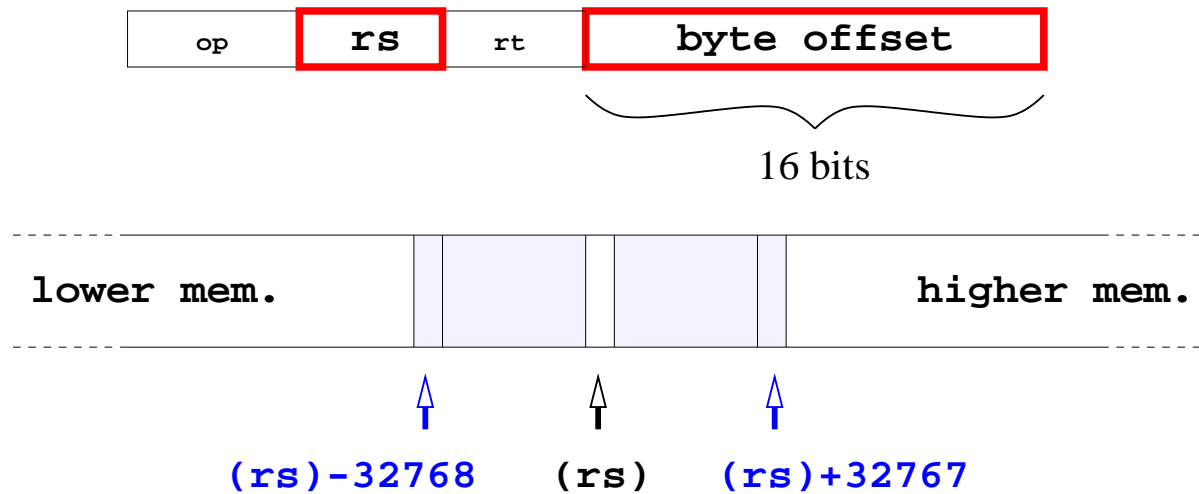


## Immediate addressing:



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

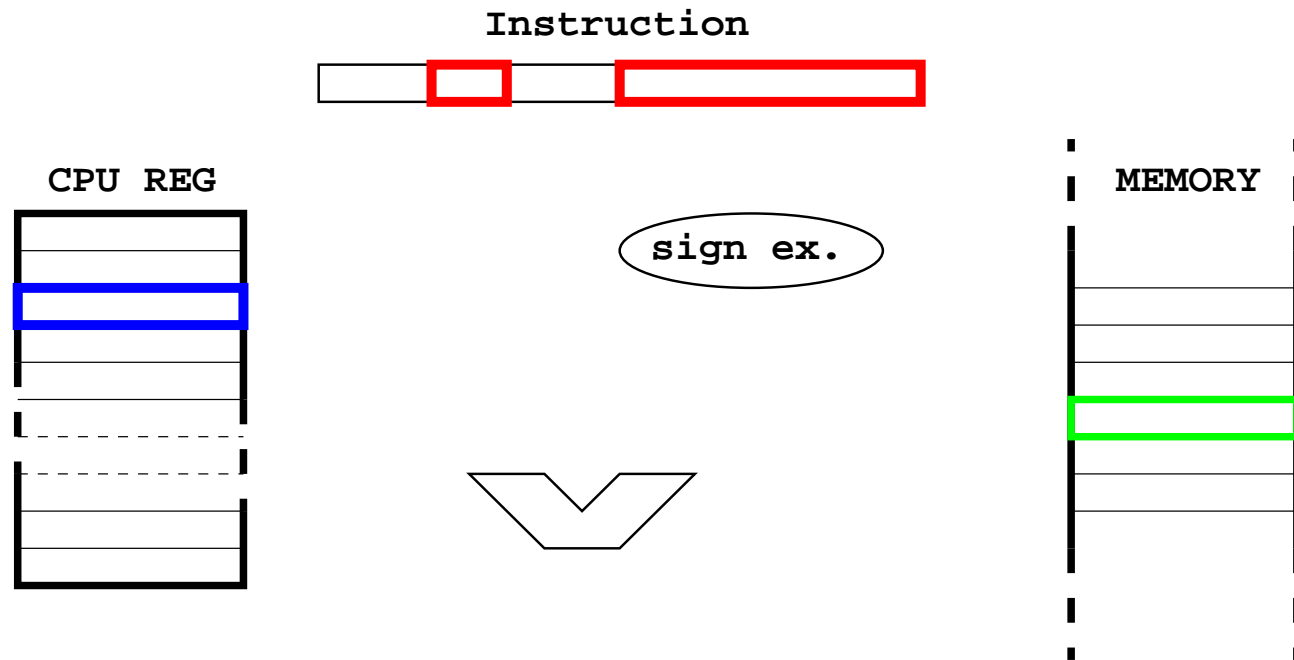
$$\text{ADDR} = \text{byte offset} + (\text{rs})$$



Example: `lw $s0, 12($s1)`

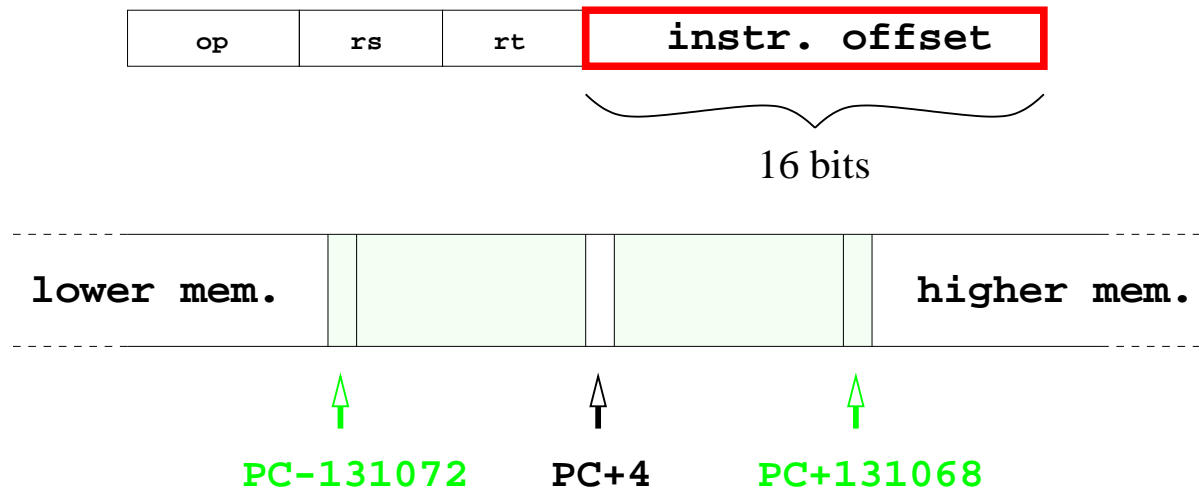


## Base addressing:



**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 ( $PC + 4$ ).

$$ADDR = offset \ll 2 + PC+4$$



## MIPS addressing modes

---

Example:

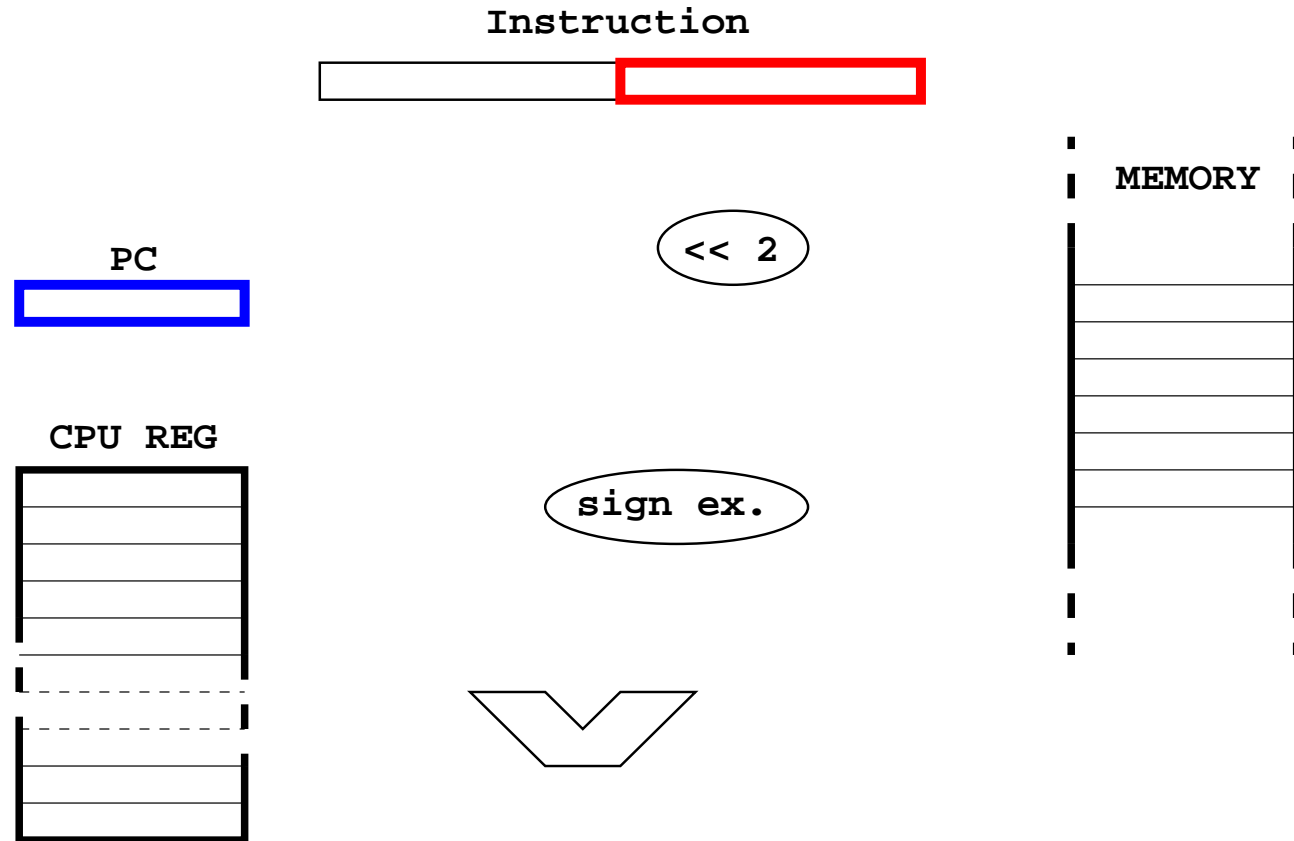
```
...
addi $s0, $s0, 1
beq  $s0, $s1, label
addi $s0, $s0, 1
addi $s0, $s0, 1
label: addi $s0, $s0, 1
addi $s0, $s0, 1
...
```

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:



**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*)



↑  
0

↑

4294967295

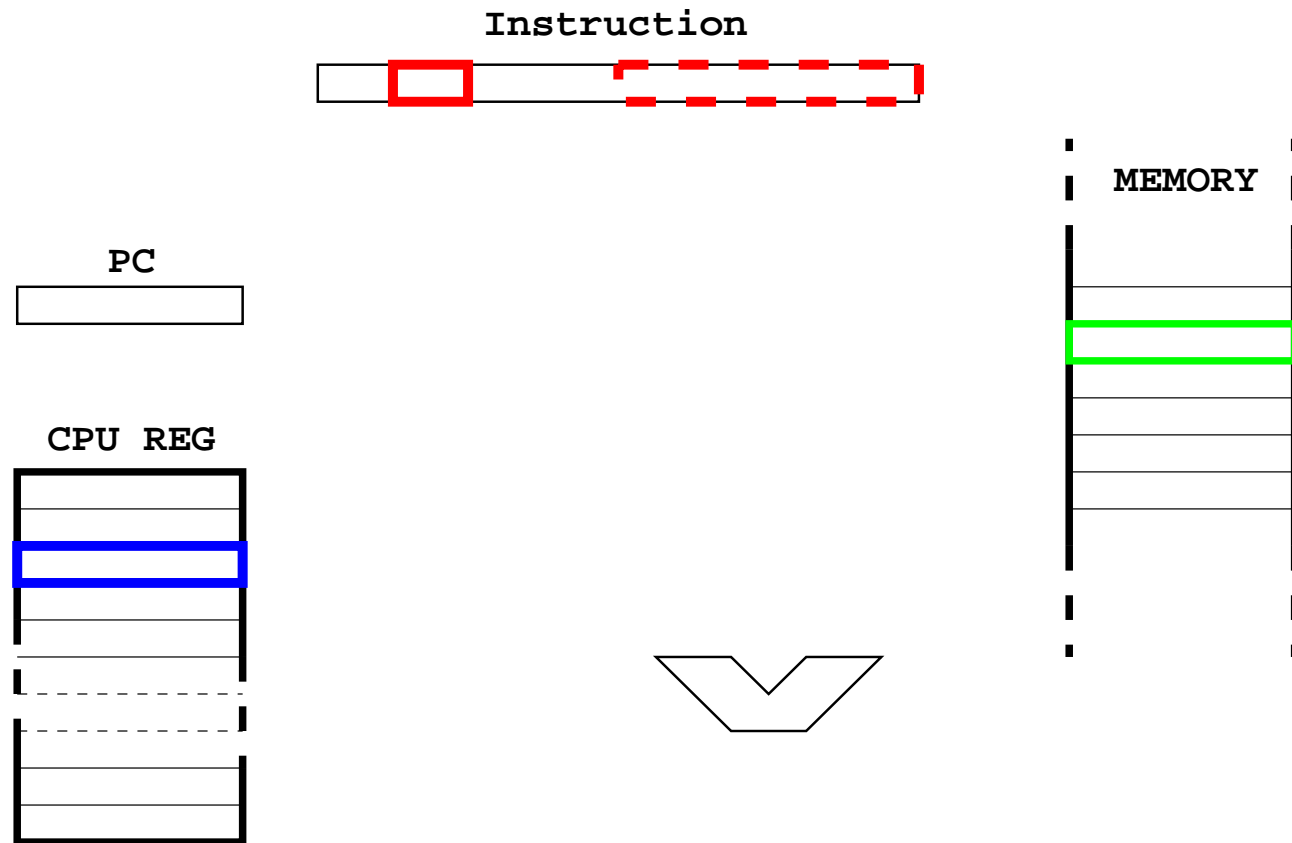
Example: `jr $ra`

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

Example: `lw $s0, 0($s1)`

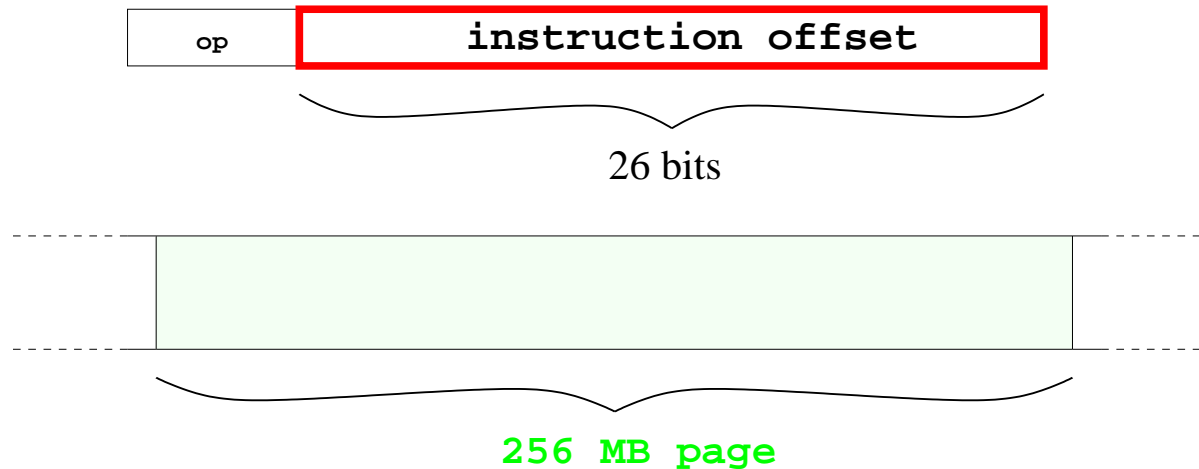


## Indirect addressing:

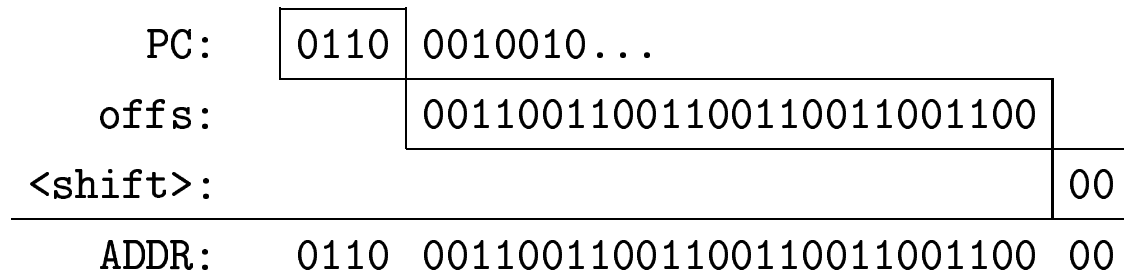


**Direct addressing:** the address is the immediate (no in MIPS)

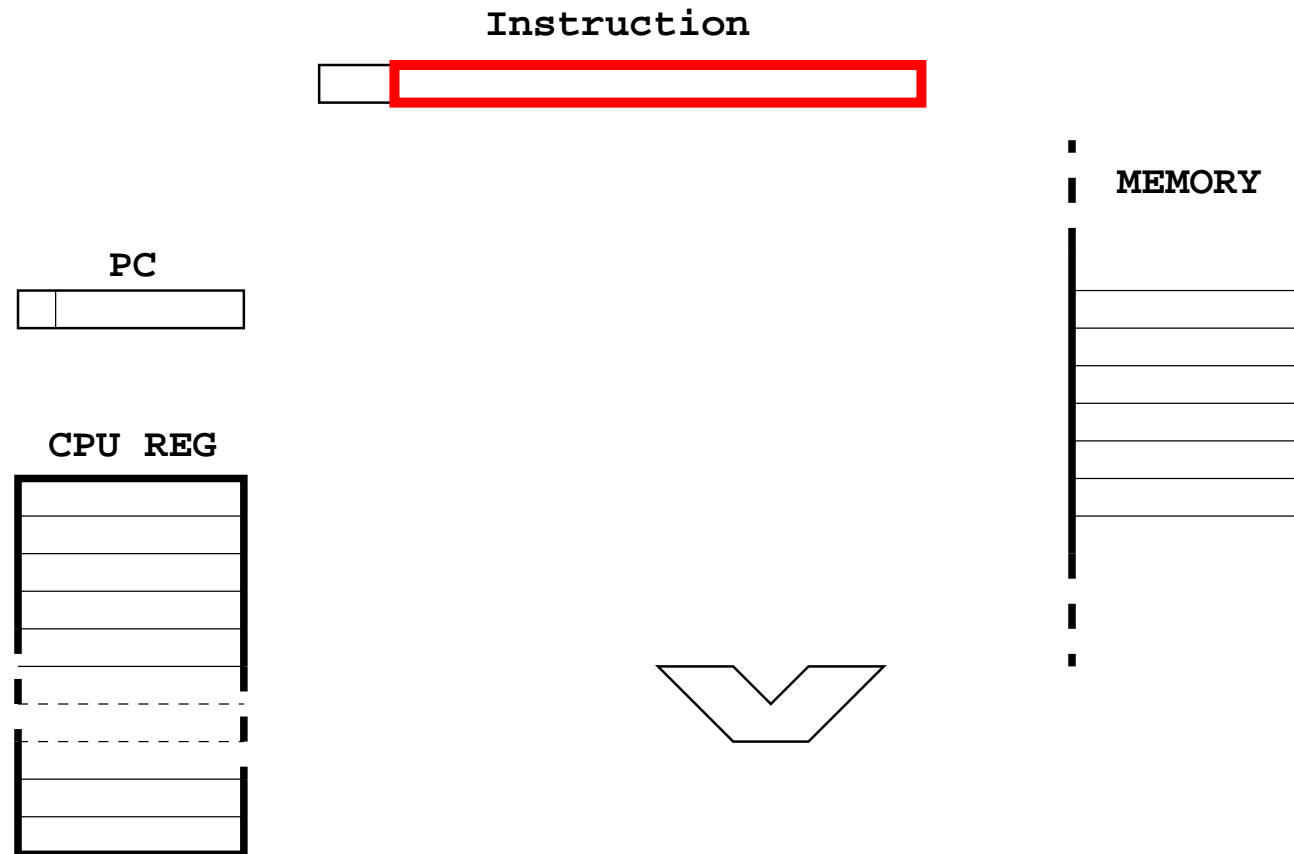
**Pseudodirect addressing:** the 26-bit immediate field of the jump instruction (`j`, `jal`) is the instruction offset within the 256 MB page given by the four most significant bits of the PC.



Example:



## Pseudo-direct addressing:



NOTE THAT:

**Addressing mode  $\neq$  instruction type (format)**

Example:

INSTRUCTION	TYPE	ADDRESSING MODE(S)
addi		
beq		
sw		



## 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)
```

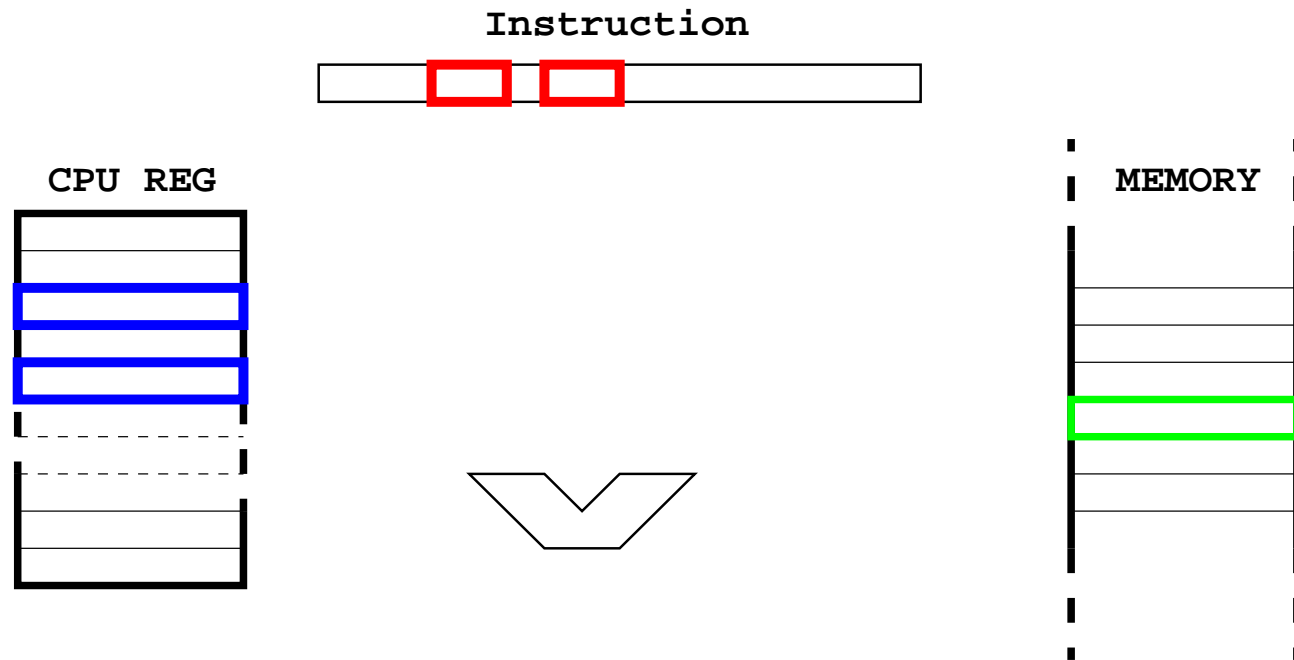
can be replaced by the single PowerPC instruction

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

- pointer offset from a pointer
- does not require additional hardware



## Indexed addressing:



**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
```

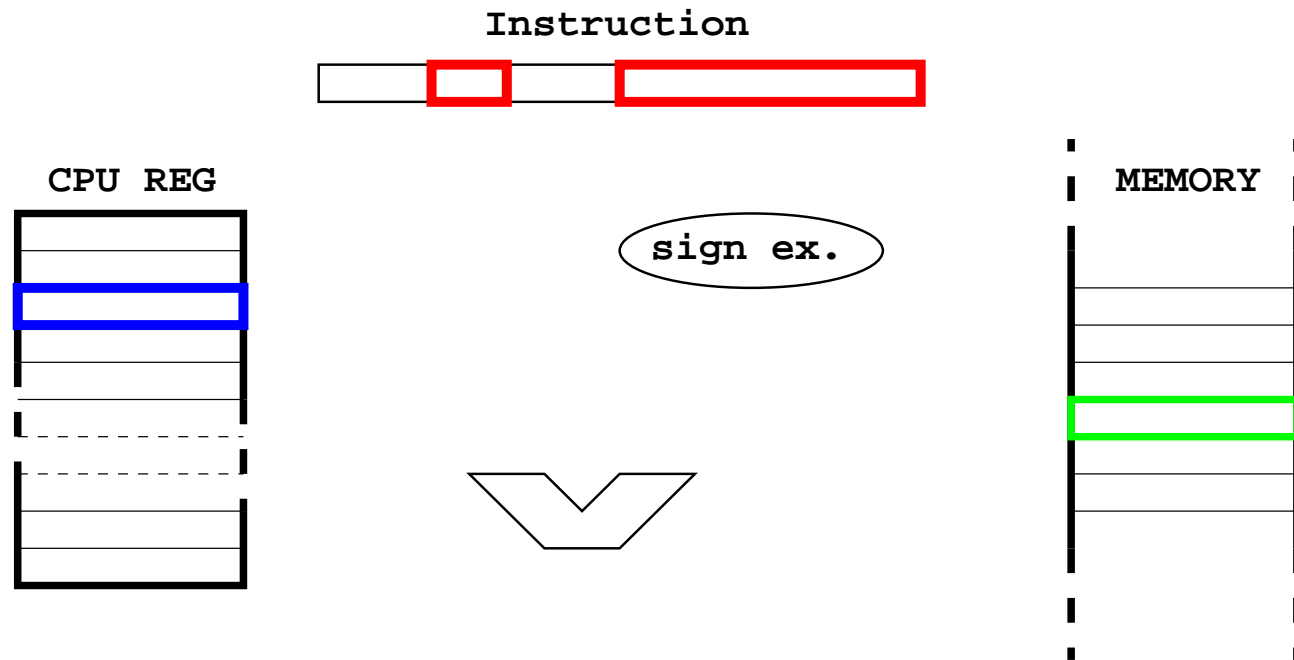
can be replaced by the single PowerPC instruction

```
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



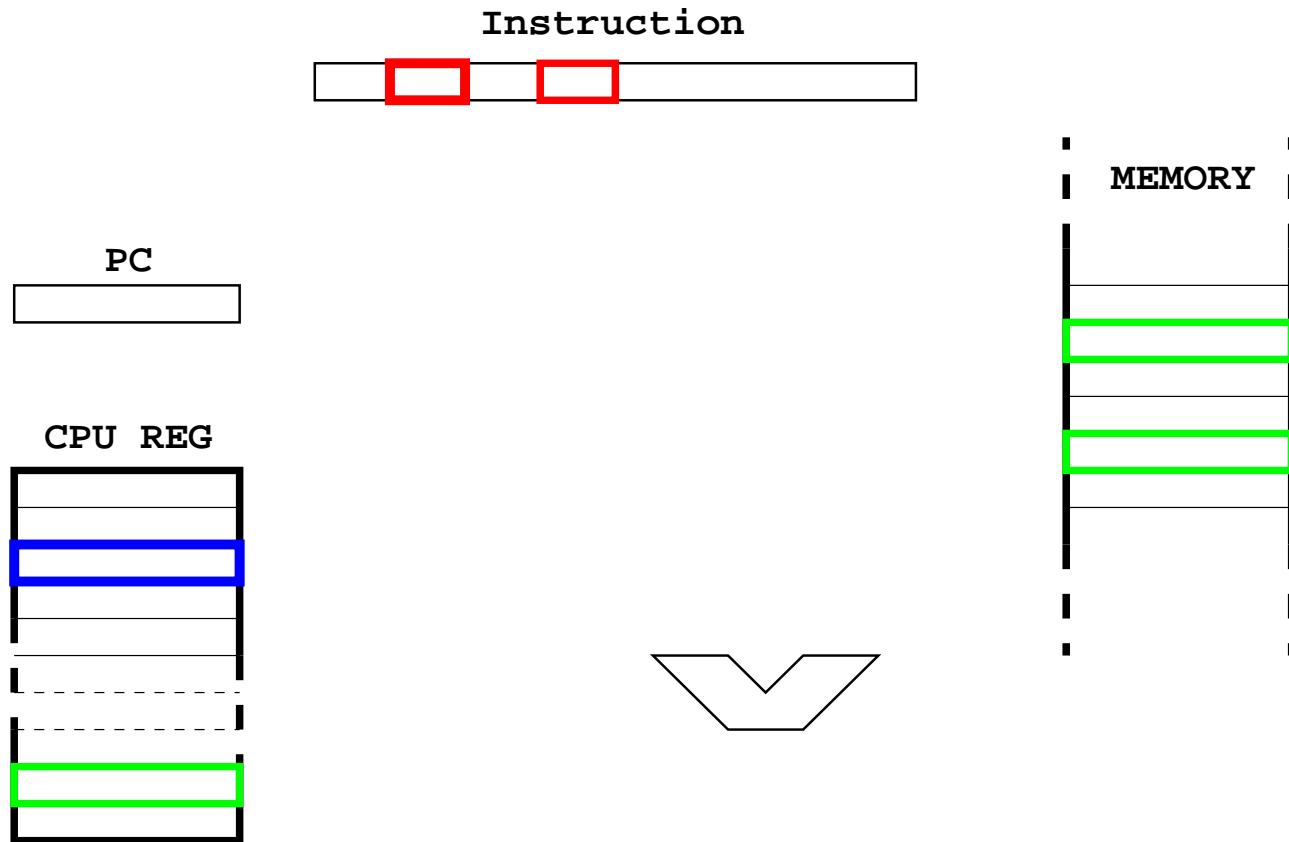
## Update addressing:



## One more addressing mode (not in PowerPC)

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

Example, PDP-11 assembly language:      `move $8, @($3)`



## Recommended exercises

- Ex. 3.12 to 3.17, 3.19, 3.20

