UCCD1133

Introduction to Computer Organisation and Architecture

Chapter 4-3

Addressing Modes

Disclaimer

- This slide may contain copyrighted material of which has not been specifically authorized by the copyright owner. The use of copyrighted materials are solely for educational purpose. If you wish to use this copyrighted material for other purposes, you must first obtain permission from the original copyright owner.

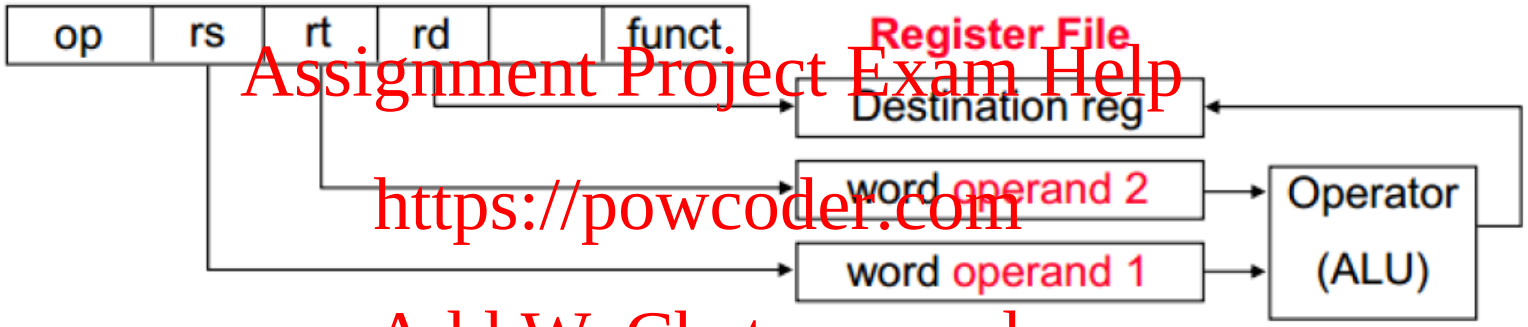
Addressing Modes

- What is addressing mode?
  - Addressing modes are the ways of specifying an operand or a memory address.
  - The address of the operand determined is called the effective address.

- Basic MIPS addressing modes:
  - Register addressing
  - Immediate addressing
  - Base (displacement) addressing
  - PC-Relative addressing
  - Pseudo-direct addressing

Register addressing

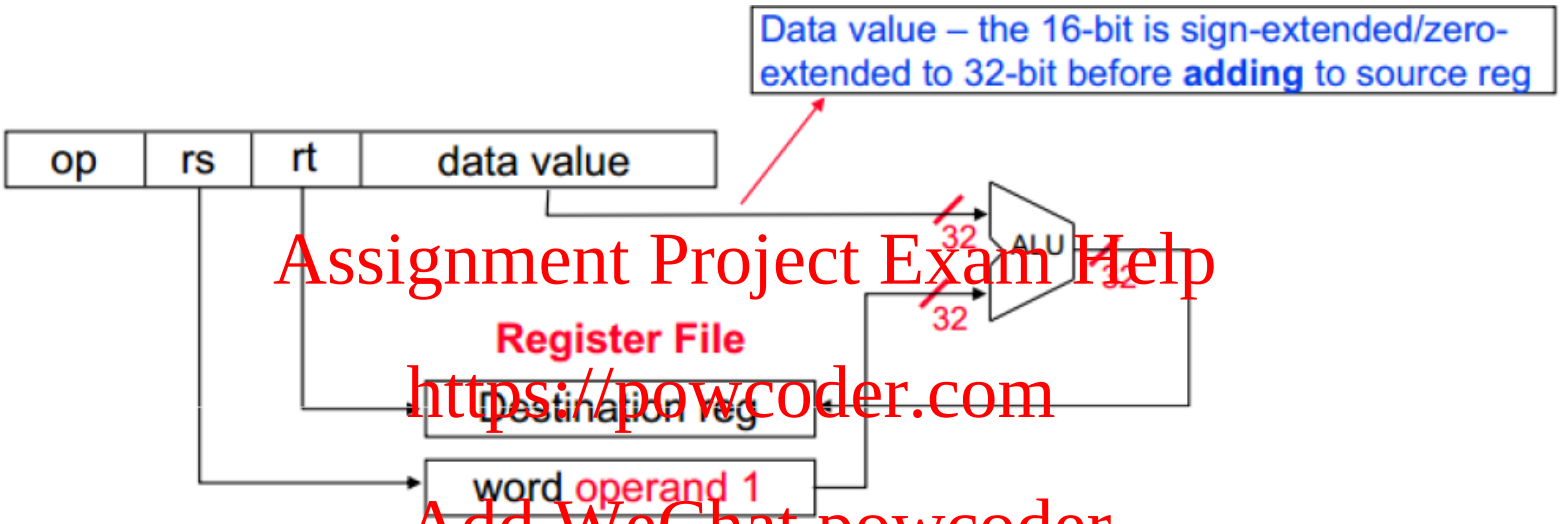
- The operand is in a register.
  - Example: `add $s0, $t2, $t3`
  - Example: `sub $t8, $s1, $0`



- Used by instructions:
  - Arithmetic instructions: add, sub, …
  - Logical instructions: and, or, sll, srl, …
  - Program control instructions: slt, sltu, jr, …

Immediate Addressing

- The operand is a 16-bit constant contained within the instruction.
  - Example: `ori  $t3, $t7, 0x00FF`

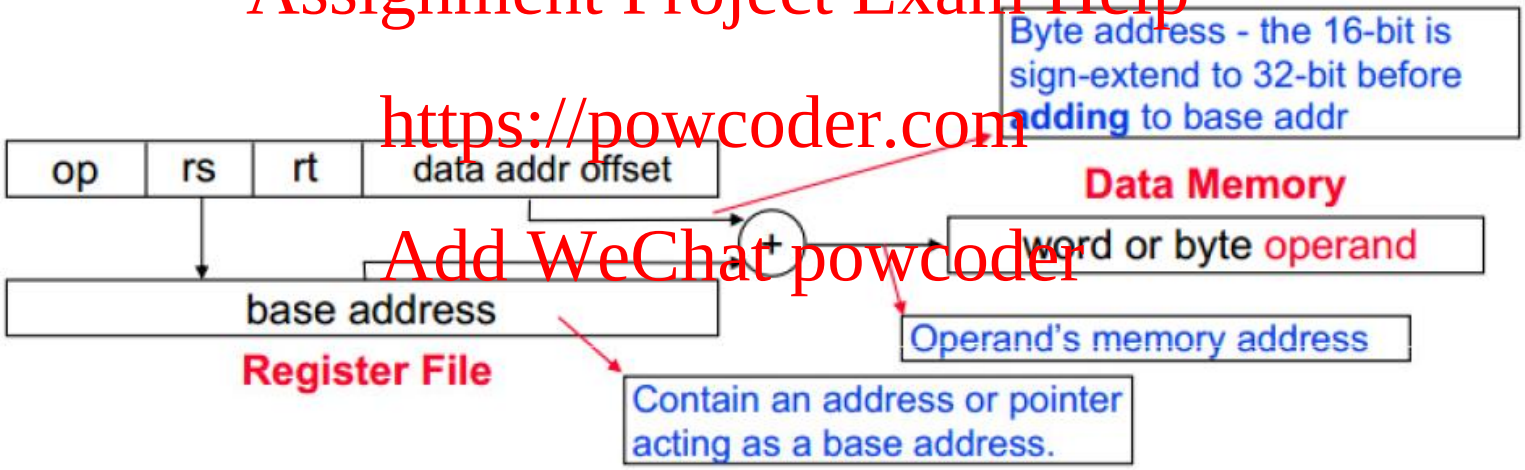


- Used by instructions:
  - Arithmetic and logical instructions: addi, andi, ori, …
  - Data transfer instructions: lui
  - Program control instructions: slti, sltiu, …

- Zero-extending 16-bit – logical immediate instructions (andi, ori, etc.)
- Sign-extending 16 bit – arithmetic and program control immediate instructions (addi, addiu, slti, sltiu, etc)

Base (displacement) addressing

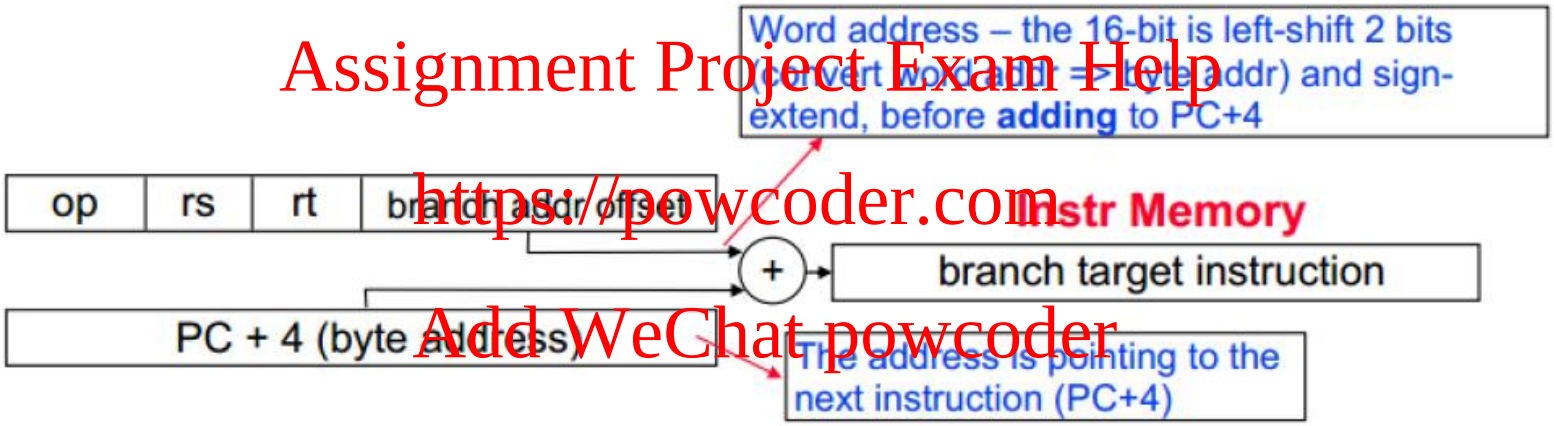
- The operand's memory address = `base address + sign-extended immediate`
  - Example: `lw  $s4, 72($0)`
  - address = `$0 + 72`
  - Example: `sw  $t2, -24($t1)`
  - address = `$t1 - 24`



- Used by instructions:
  - Data transfer instructions: lw, sw, lb, sb, …

PC-Relative Addressing

- The new instruction's memory address is obtained by summing the PC and a 16-bit constant contained within the current instruction.
- Used by instructions:
  - Branch instructions: beq, bne



PC-Relative Addressing

- The new instruction's memory address is obtained by summing the PC and a 16-bit constant contained within the current instruction.

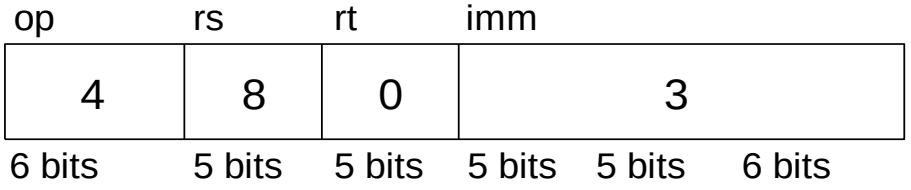
  - Example:

```
0x10            beq   $t0, $0, else
0x14            addi  $v0, $0, 1
0x18            addi  $sp, $sp, i
0x1C            jr    $ra
0x20      else: addi  $a0, $a0, -1
0x24            jal   factorial
```


Assembly Code

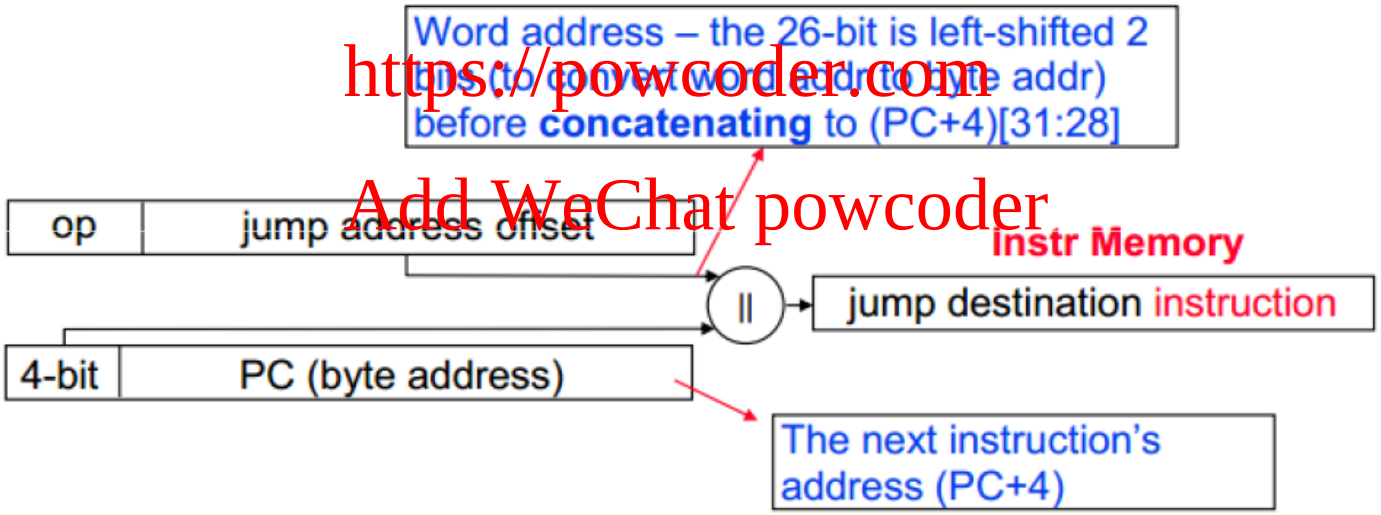
```
beq $t0, $0, else
(beq $t0, $0, 3)
```

Field Values



Pseudo-direct Addressing

- The new instruction's address (jump target address) in memory is obtained by concatenating the left-shifted 26-bit constant contained within the current instruction with the upper 4-bits of the PC.
- Used by instructions:
  - Jump instructions: j, jal.



Pseudo-direct Addressing

- The new instruction's address (jump target address) in memory is obtained by concatenating the left-shifted 26-bit constant contained within the current instruction with the upper 4-bits of the PC.

  - Example:

```
0x0040005C          jal   sum
...
0x004000A0  sum:    add   $v0, $a0, $a1
```


JTA 0000 0000 0100 0000 0000 0000 1010 0000 (0x004000A0)

26-bit addr 0000 0000 0100 0000 0000 0000 1010 0000 (0x0100028)

0 1 0 0 0 2 8

Field Values

op	imm
3	0x0100028
6 bits	26 bits

Machine Code

op	addr	
000011	00 0001 0000 0000 0000 0010 1000	(0x0C100028)
6 bits	26 bits	

MIPS Memory Map

- What is stored in memory?
  - Instructions (also called text)
  - Data
    - Global/static: allocated before program begins
    - Dynamic: allocated within program
- How big is memory?
  - At most 232 = 4 gigabytes (4 GB)
  - From address 0x00000000 to 0xFFFFFFFF