

MIPS

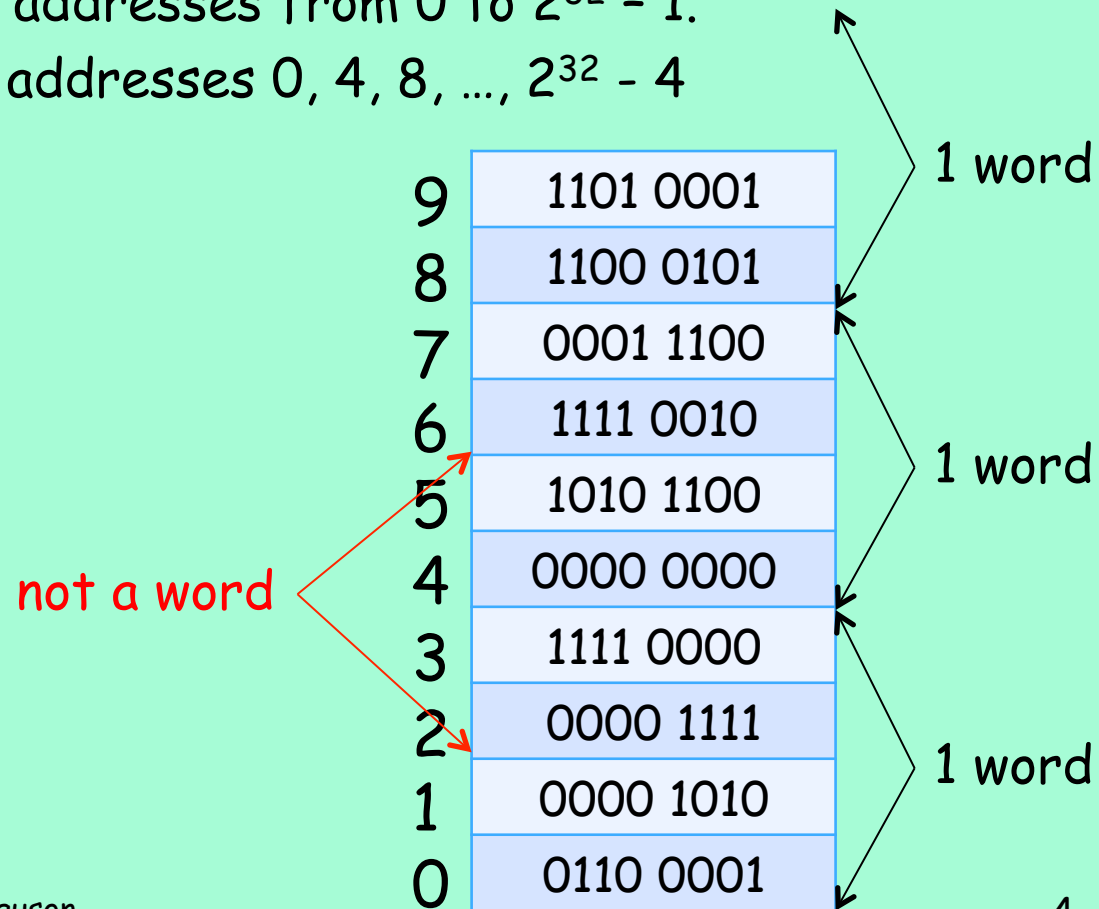
Addressing Modes and Memory Architecture

(Second Edition: Section 3.8
Fourth Edition: Section 2.10)
from Dr. Andrea Di Blas' notes



Memory Organization and Addressing

- Memory may be viewed as a single-dimensional array of individually addressable bytes. 32-bit words are **aligned** to 4 byte boundaries.
 - 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

Ex: 0000 0001 0010 0011 0100 0101 0110 0111

x..x1101		Little Endian
x..x1100		
x..x0111	00000001	
x..x0110	00100011	
x..x0101	01000101	
x..x0100	01100111	
x..x0011		
x..x0010		

x..x1101		Big Endian
x..x1100		
x..x0111	01100111	
x..x0110	01000101	
x..x0101	00100011	
x..x0100	00000001	
x..x0011		
x..x0010		



MIPS addressing modes

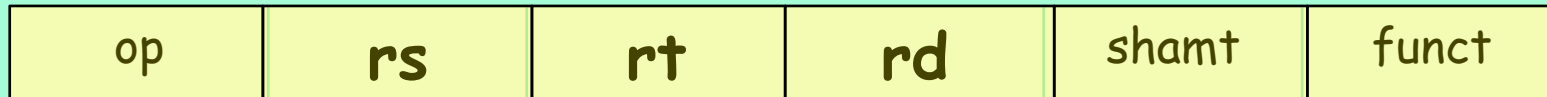
Addressing modes are the ways of specifying an operand or a memory address.

- Register addressing
- Immediate addressing
- Base addressing
- PC-relative addressing
- Indirect addressing
- Direct addressing (almost)

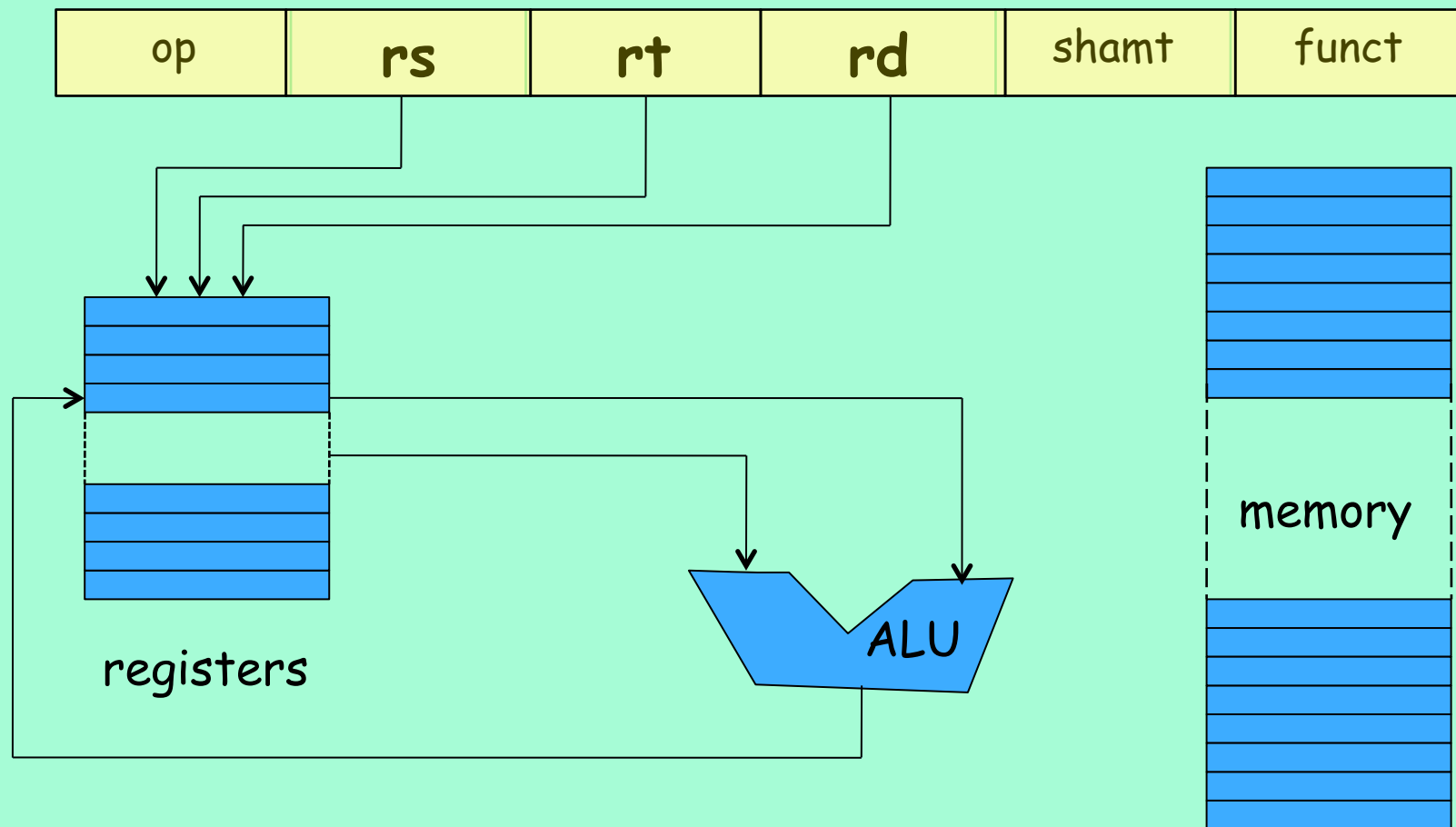


Register addressing

- Operands are in a register.
- Example: add \$3, \$4, \$5
- Takes n bits to address 2^n registers

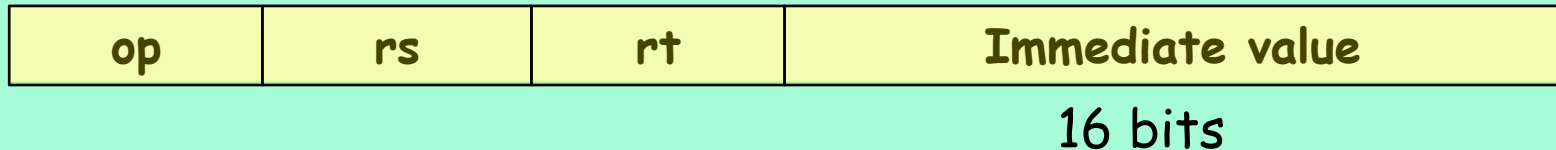


Register addressing



Immediate Addressing

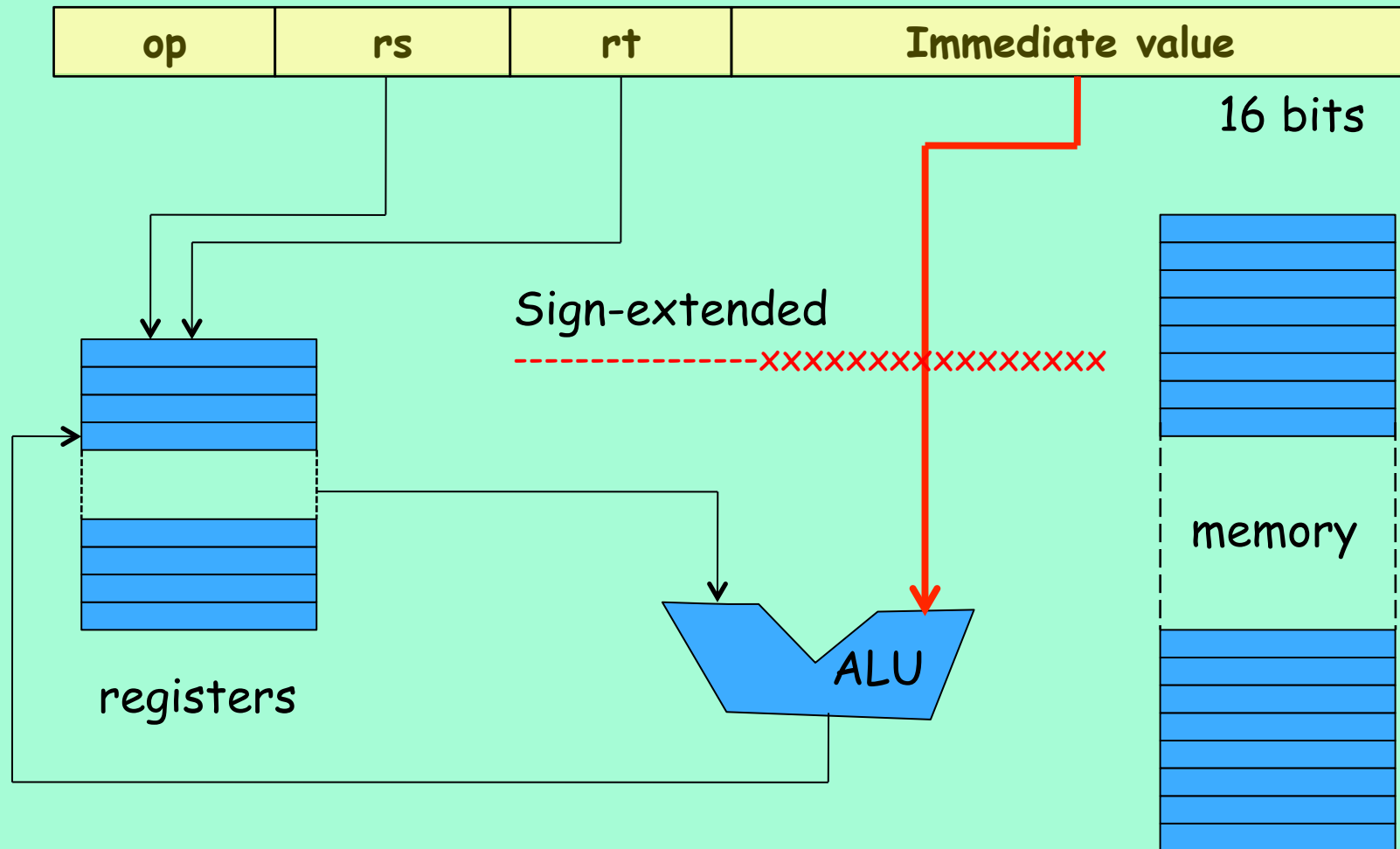
- The operand is embedded inside the encoded instruction.



16 bit two's-complement number:
 $-2^{15} - 1 = -32,769 < \text{value} < +2^{15} = +32,768$



Immediate addressing



Example is addi or similar



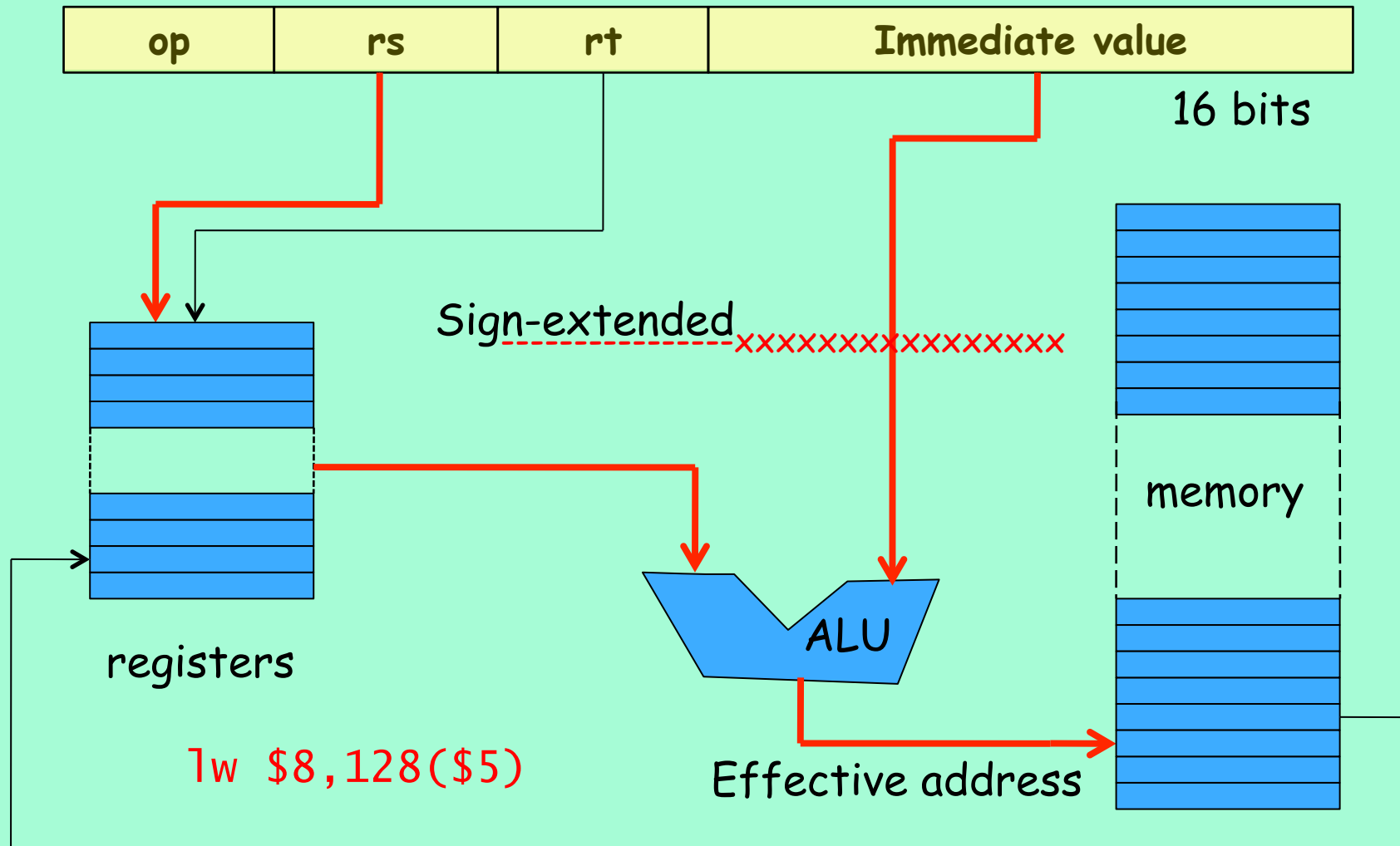
Base (or Base-offset or displacement) Addressing

- The address of the operand is the sum of the immediate and the value in a register (rs).
- 16-bit immediate is a two's complement number
- Ex: lw \$15,16(\$12)

op	rs	rt	Immediate value
----	----	----	-----------------



Base addressing



PC-relative addressing: the value in the immediate field is interpreted as an offset of the next instruction ($PC+4$ of current instruction)

Example: `beq $0,$3,Label`

op	rs	rt	Immediate value
----	----	----	-----------------



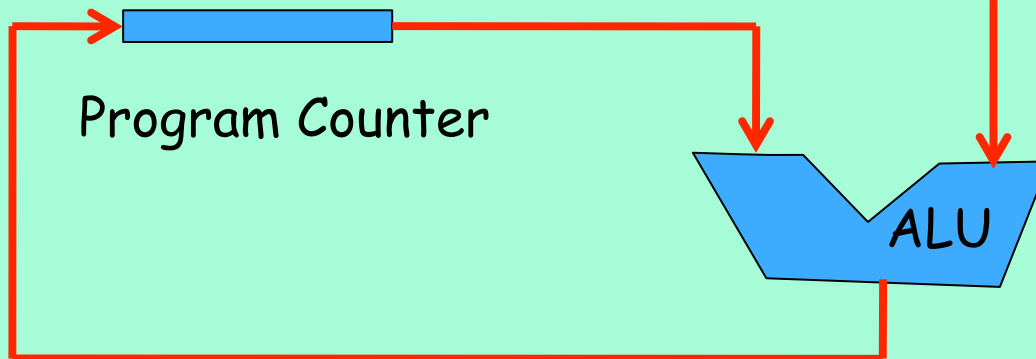
PC-relative addressing



16 bits

Shifted by 2 and Sign-extended

-----xxxxxxxxxxxxxx00



beq \$0,\$5,Label



Detail of MIPS PC-Relative

address	instruction
40000008	addi \$5, \$5, 1
4000000C	beq \$0, \$5, label
40000010	addi \$5, \$5, 1
40000014	addi \$5, \$5, 1
40000018	label addi \$5, \$5, 1
4000001C	addi \$5, \$5, 1
40000020	etc...

Binary code to beq \$0,\$5, label is 0x10050002, which means 2 instructions from the next instruction.

PC = 0x4000000C
 PC+4 = 0x40000010
 Add 4*2 = 0x00000008
 Eff. Add. = 0x40000018

op	rs	rt	Immediate value
00010	00000	00101	000000000000000010



Register Direct Addressing: the value the (memory) effective address is in a register. Also called "Indirect Addressing".

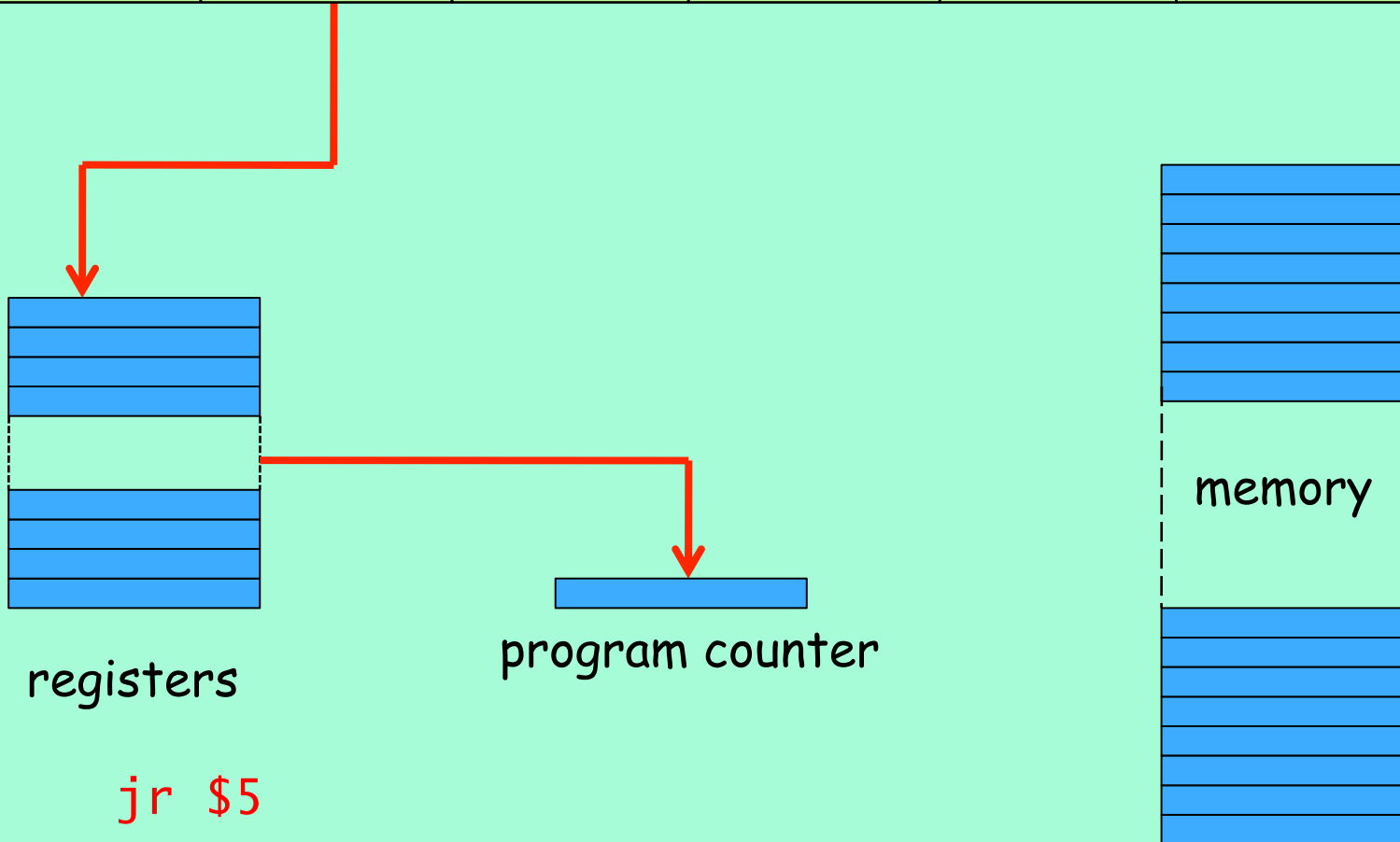
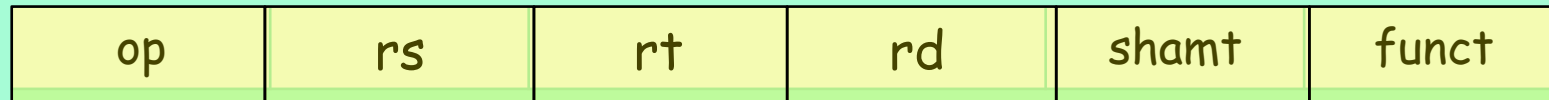
Special case of base addressing where offset is 0.
Used with the jump register instructions (jr, jalr).

Example: jr \$31

op	rs	rt	rd	shamt	funct
000000	rs	00000	00000	00000	001000



Register Direct



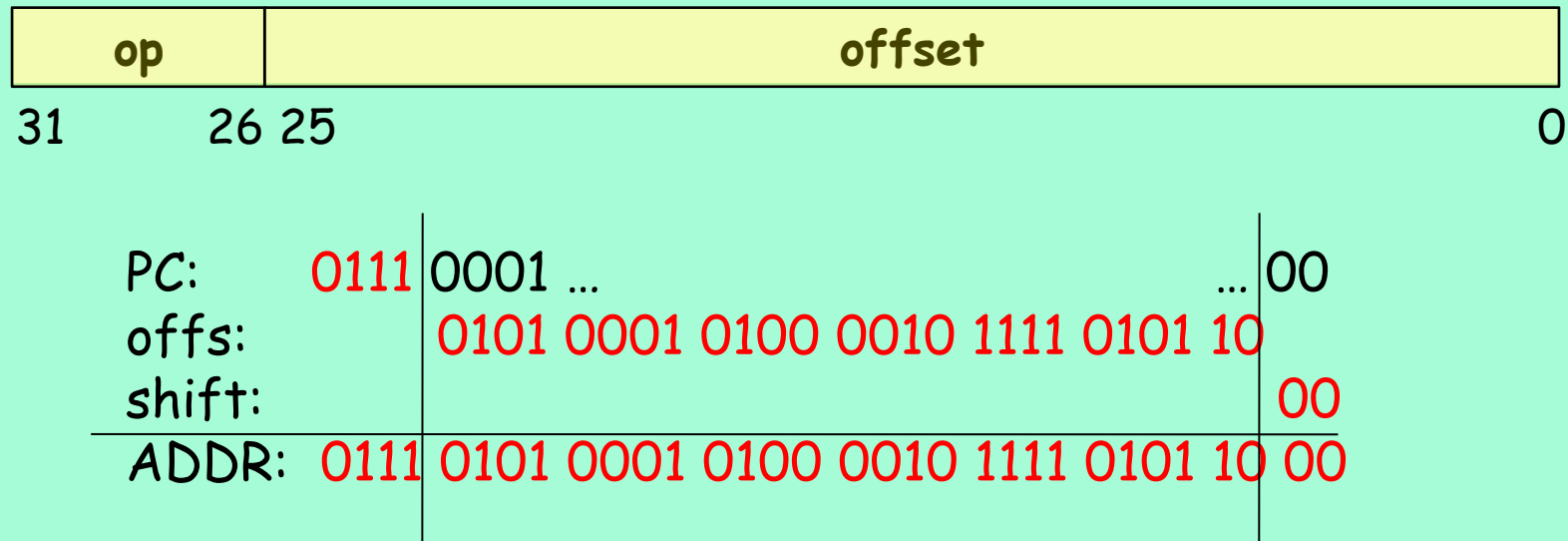
jr \$5



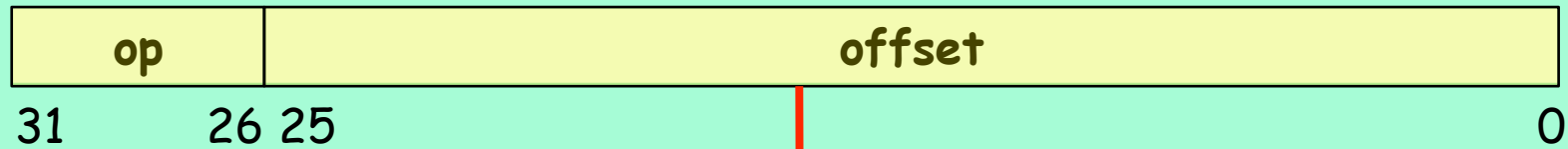
Direct Addressing: the address is "the immediate". 32-bit address cannot be embedded in a 32-bit instruction.

Pseudodirect addressing: 26 bits of the address is embedded as the immediate, and is used as the instruction offset within the current 256MB (64MWord) region defined by the MS 4 bits of the PC.

Example: j Label



Pseudodirect addressing



j Label



program counter



Caution: Addressing mode is not Instruction type

- Addressing mode is how an address (memory or register) is determined.
- Instruction type is how the instruction is put together.
- Example: addi, beq, and lw are all I-types instructions. But
 - addi uses immediate addressing mode (and register)
 - beq uses pc-relative addressing (and register)
 - lw uses base addressing (and register)



MIPS Addressing Modes

1. REGISTER: a source or destination operand is specified as content of one of the registers \$0-\$31.
2. IMMEDIATE: a numeric value embedded in the instruction is the actual operand.
3. PC-RELATIVE: a data or instruction memory location is specified as an offset relative to the incremented PC.
4. BASE: a data or instruction memory location is specified as a signed offset from a register.
5. REGISTER-DIRECT: the value the effective address is in a register.
6. PSEUDODIRECT: the memory address is (mostly) embedded in the instruction.



PowerPC and x86 addressing modes and instructions

- **PowerPC:** 2nd edition: pp. 175-177, 4th edition: Appendix E.
- **80x86:** 2nd edition: pp. 177-185, 4th edition: Section 2.17.



Additional PowerPC addressing modes - 1

Indexed Addressing: The address is the sum of two registers. (note indexed addressing is different here than usually used)

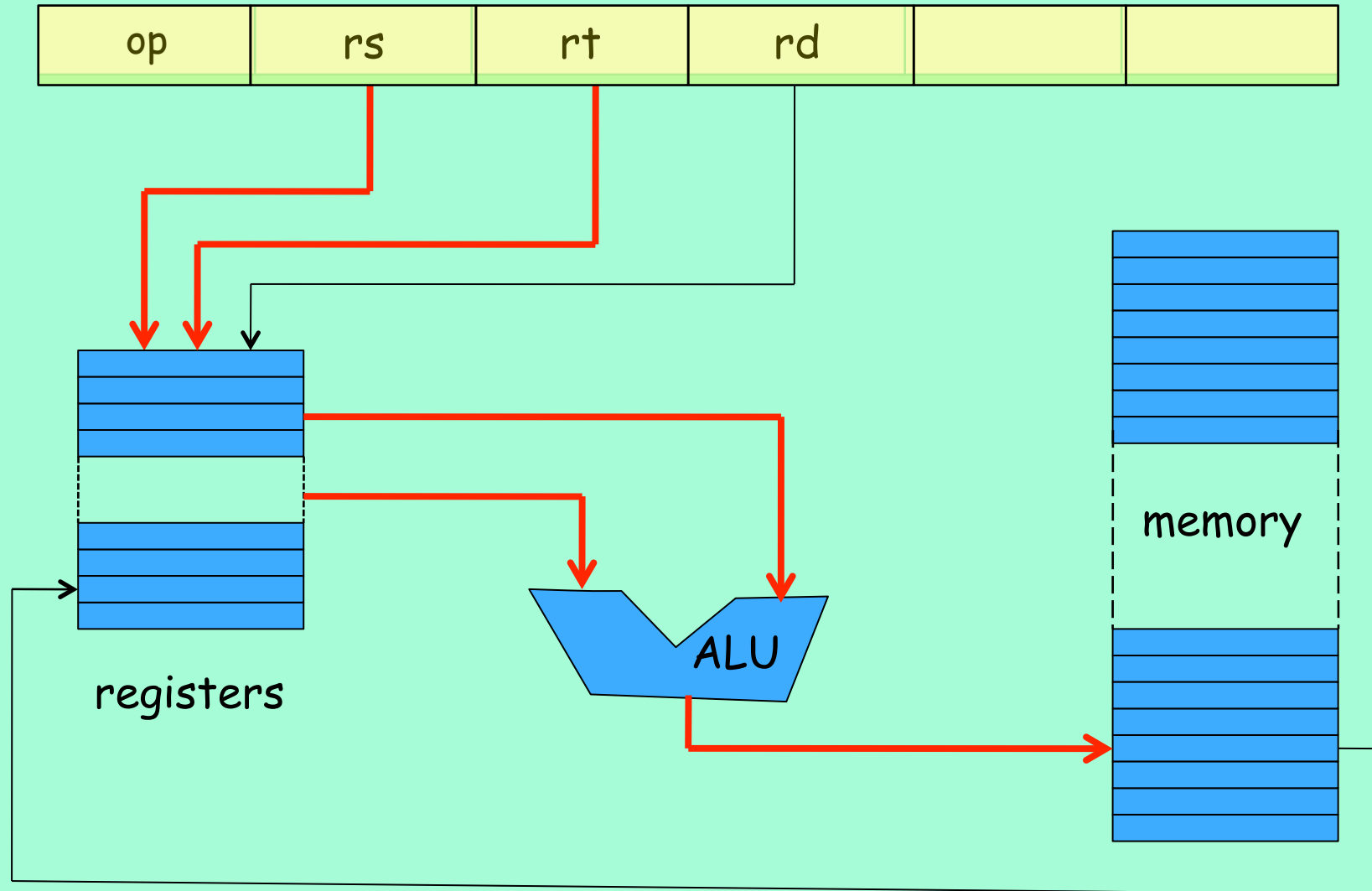
MIPS code: `add $10, $20, $13` ;\$20 is base,\$13 is index
`lw $5, 0($10)`

PowerPC: `lw $5, $20+$13` ; $\$5 \leftarrow (\$20 + \$13)$

Saves instruction for incrementing array index.
No extra hardware.



PowerPC: Indexed Addressing



Additional PowerPC addressing mode - 2

Update Addressing: base addressing with automatic base register increment.

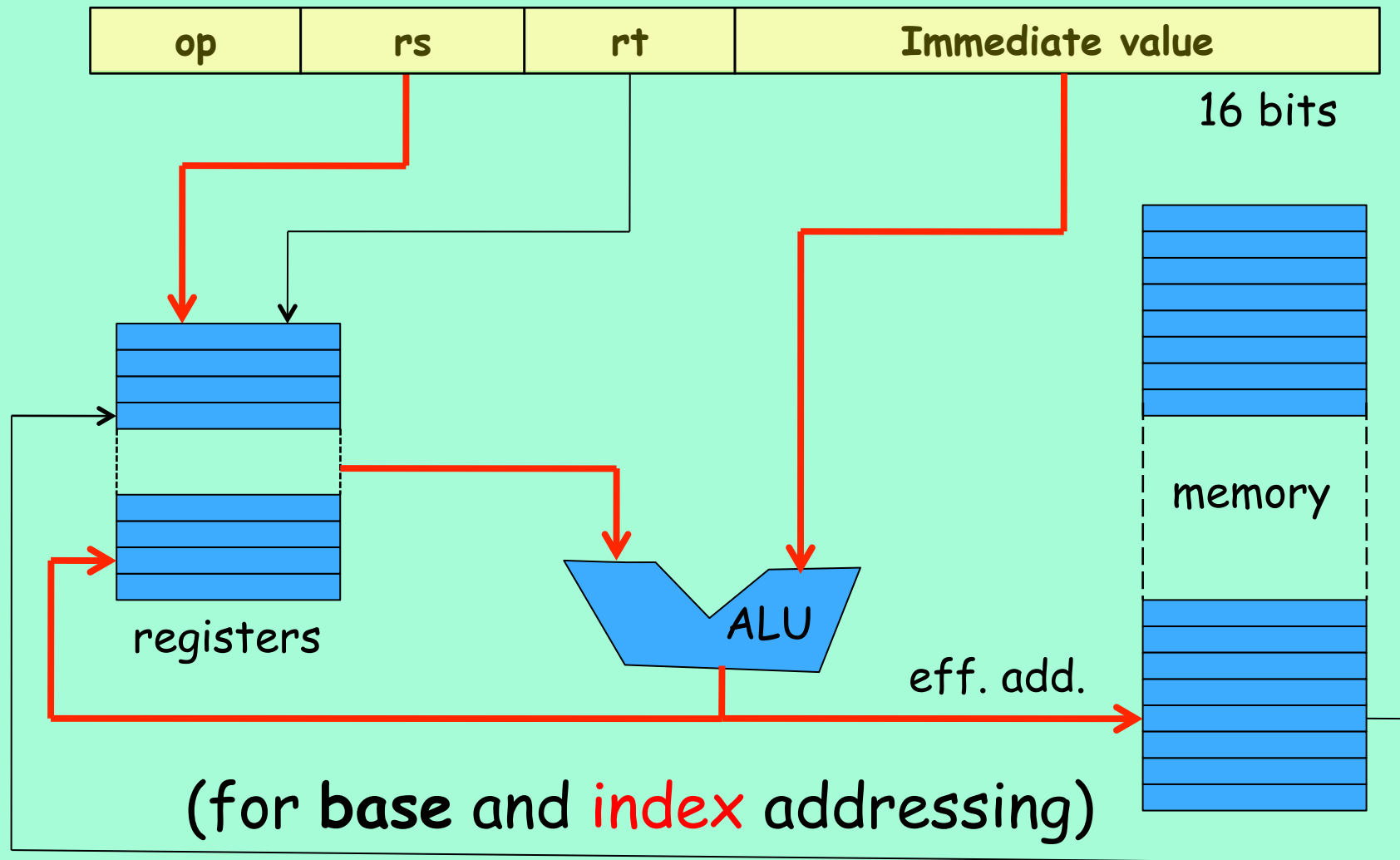
MIPS code: `lw $10, 4($13)` ; $\$10 \leftarrow \text{Mem}[\$10+4]$
`addi $13, $13, 4` ; $\$13 \leftarrow \$13+4$

PowerPC: `lwu $10, 4($13)` ; $\$10 \leftarrow \text{Mem}[\$10+4]$
; and $\$13 \leftarrow \$13+4$

Requires that two registers be written at the same time
→ more hardware.



PowerPC: Update Addressing



Additional non-RISC addressing mode

Memory Indirect Addressing: read effective address from memory. (Usually PC-relative addressing is used to get the effective address from memory).

RISC code: `lw $10, 0($13)`
`lw $5, 0($10)`

CISC: `ldi $5, Label` ; $\$5 \leftarrow \text{Mem}[\text{Label}]$

Requires two sequential data memory accesses.



CISC: Memory Indirect Addressing

