جامعة المنارة الخاصة
كلـــــية الهندســـــة
هندسة ميـــكاترونيك

المعالجات الصغرية ولغة التجميع

المحاضرة الرابعة

مدرس المقرر
د. بسام حسن

2025_2026

جَامعة المَنارة
MANARA UNIVERSITY

مفردات من المحاضرة الرابعة :

- تحويل تعليمات لغة MIPS إلى تعليمات الآلة الموافقة
- أنماط العنونة في MIPS

تحويل تعليمات لغة MIPS إلى تعليمات الآلة الموافقة

- All MIPS instructions are 32 bits long
- Basic Instruction Formats
  - R-Type (Registers)
  - I-Type (Immediate and data transfer instructions)
  - J-Type
- Each format is assigned a distinct set of values in the first field (op or opcode)
  - The hardware knows how to treat the last half of the instruction

Machine language: binary representation of instructions

- All instructions are encoded in 4 bytes --- 32 bits

R-Type

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

R-Type

- **Register-type, 3 register operands:**
  - **rs, rt:** source registers
  - **rd:** destination register
- **Other fields:**
  - **op:** the operation code or opcode (0 for R-type instructions)
  - **funct:** the function together, the opcode and function tell the computer what operation to perform
  - **shamt:** the shift amount for shift instructions, otherwise it's 0

R-Type-Example 1

نريد تحويل التعليمة add $s0,$s3,$t0 إلى لغة الآلة:

سنحتاج الى الجداول التالية

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
$zero	0	The Constant Value 0	N.A.
$at	1	Assembler Temporary	No
$v0-$v1	2-3	Values for Function Results and Expression Evaluation	No
$a0-$a3	4-7	Arguments	No
$t0-$t7	8-15	Temporaries	No
$s0-$s7	16-23	Saved Temporaries	Yes
$t8-$t9	24-25	Temporaries	No
$k0-$k1	26-27	Reserved for OS Kernel	No
$gp	28	Global Pointer	Yes
$sp	29	Stack Pointer	Yes
$fp	30	Frame Pointer	Yes
$ra	31	Return Address	Yes

ارقام المسجلات

ارقام المسجلات

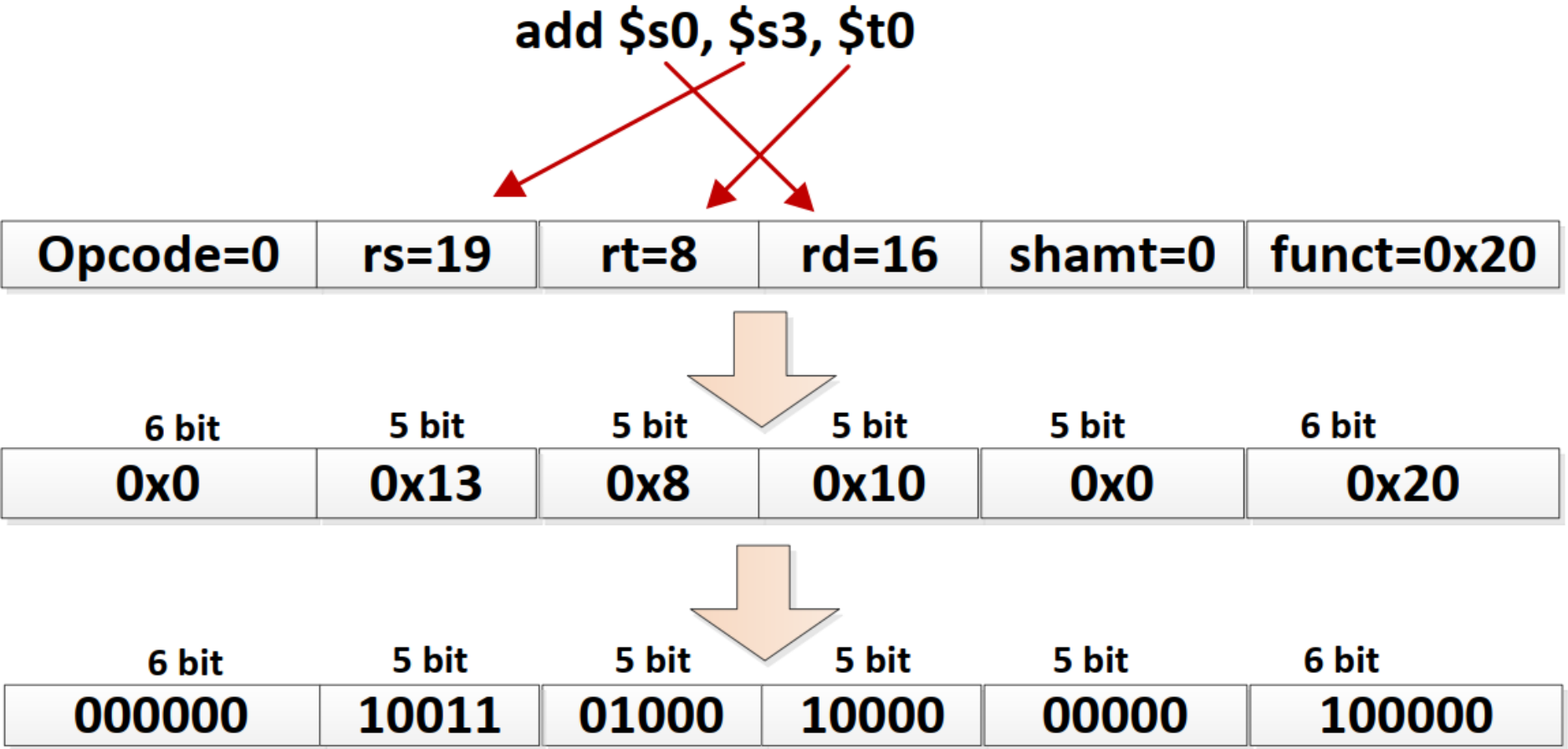
CORE INSTRUCTION SET

NAME, MNEMONIC		FOR-MAT	OPERATION (in Verilog)		OPCODE / FUNCT (Hex)
Add	add	R	R[rd] = R[rs] + R[rt]	(1)	$0 / 20_{hex}$
Add Immediate	addi	I	R[rt] = R[rs] + SignExtImm	(1,2)	8_{hex}
Add Imm. Unsigned	addiu	I	R[rt] = R[rs] + SignExtImm	(2)	9_{hex}

R-Type-Example 1

تحويل تعليمات لغة MIPS إلى تعليمات الآلة الموافقة

نريد تحويل التعليمة
add $s0,$s3,$t0 إلى لغة الآلة:

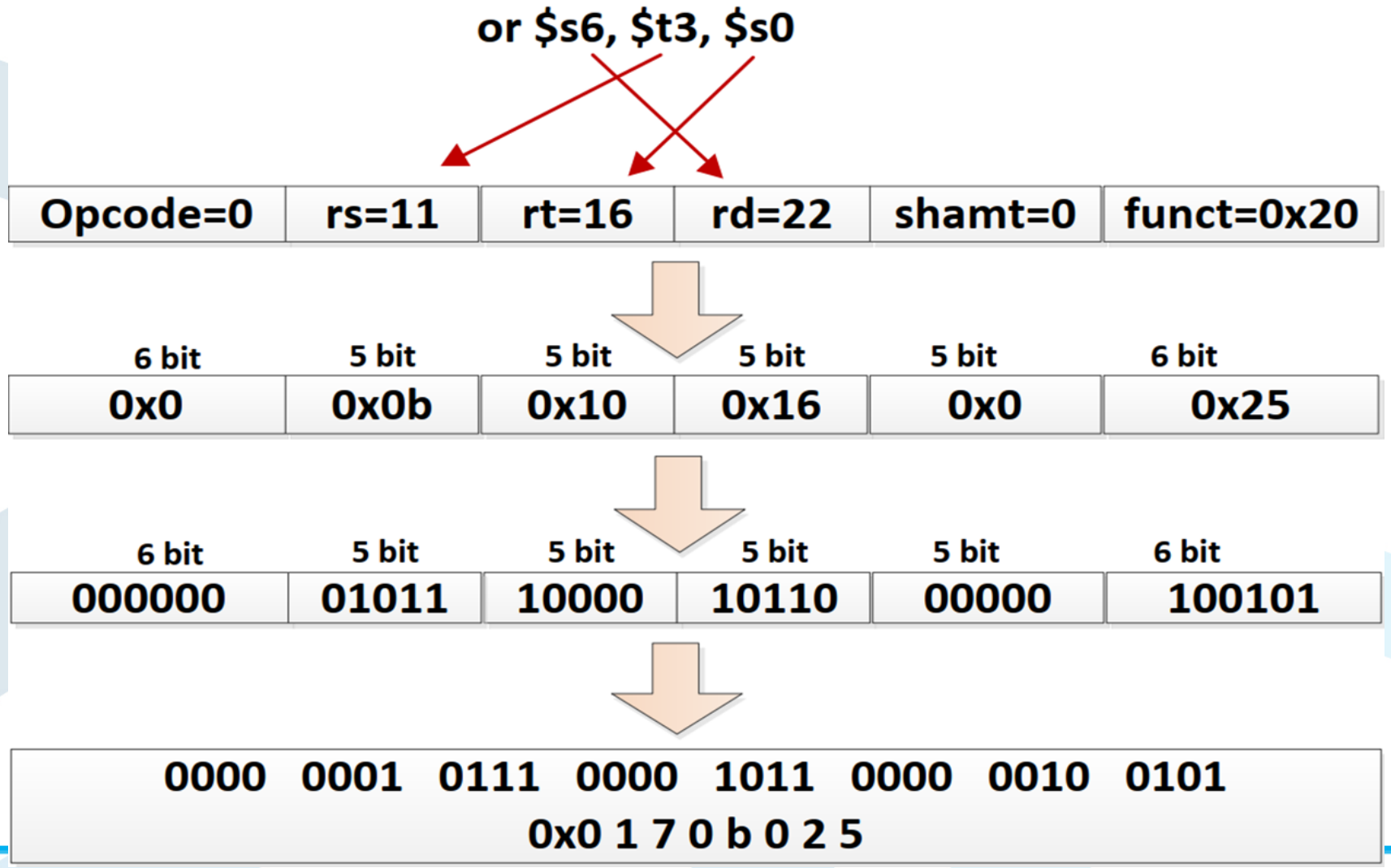


الجواب

$(0000\ 0010\ 0110\ 1000\ 1000\ 0000\ 0010\ 0000)_2$=**0x02688020**

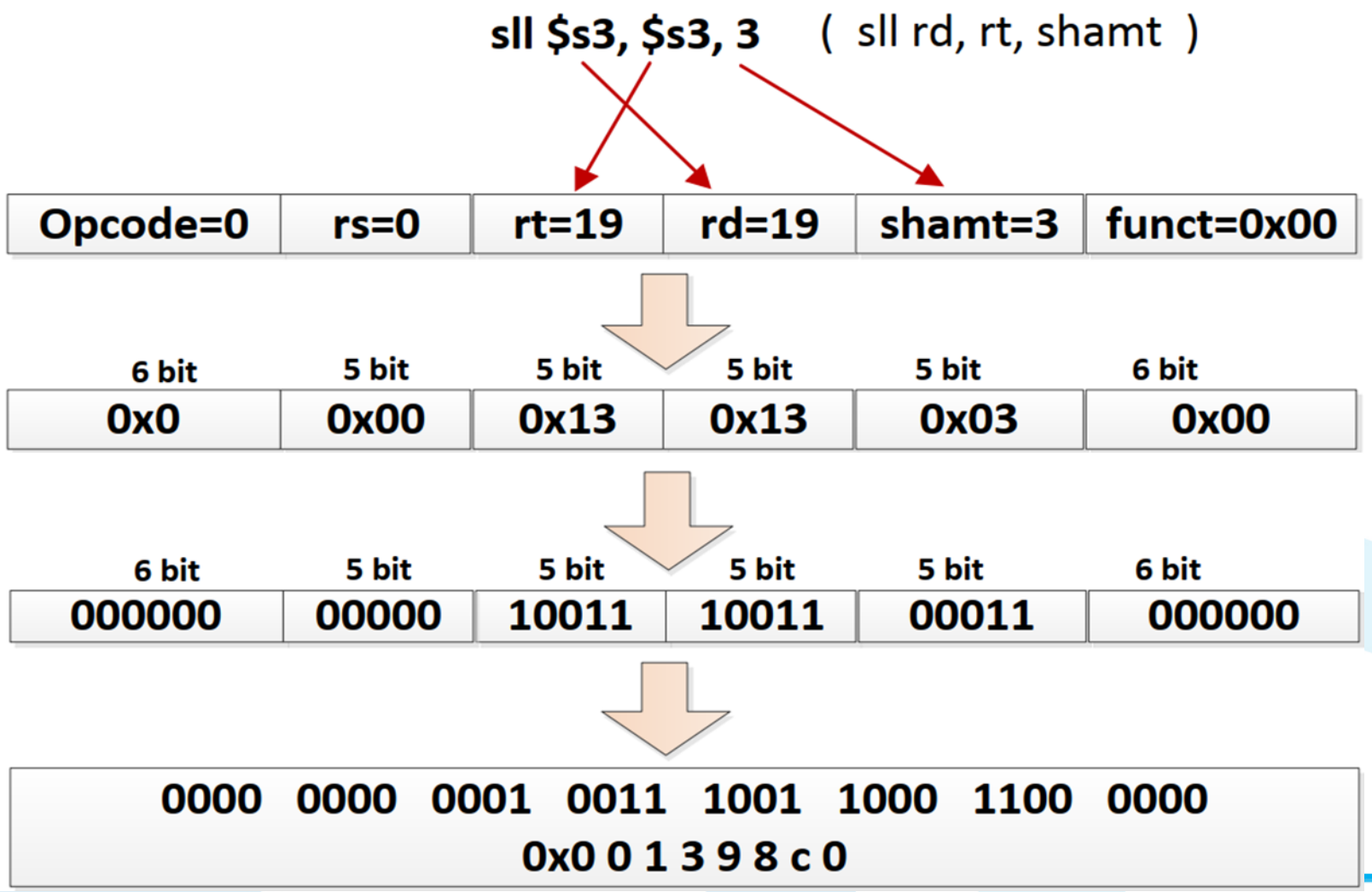
R-Type-Example 2

تحويل تعليمات لغة MIPS إلى تعليمات الآلة الموافقة

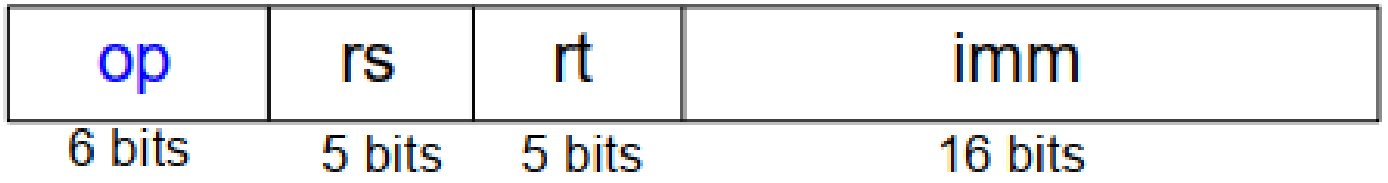


R-Type-Example 3

تحويل تعليمات لغة MIPS إلى تعليمات الآلة الموافقة



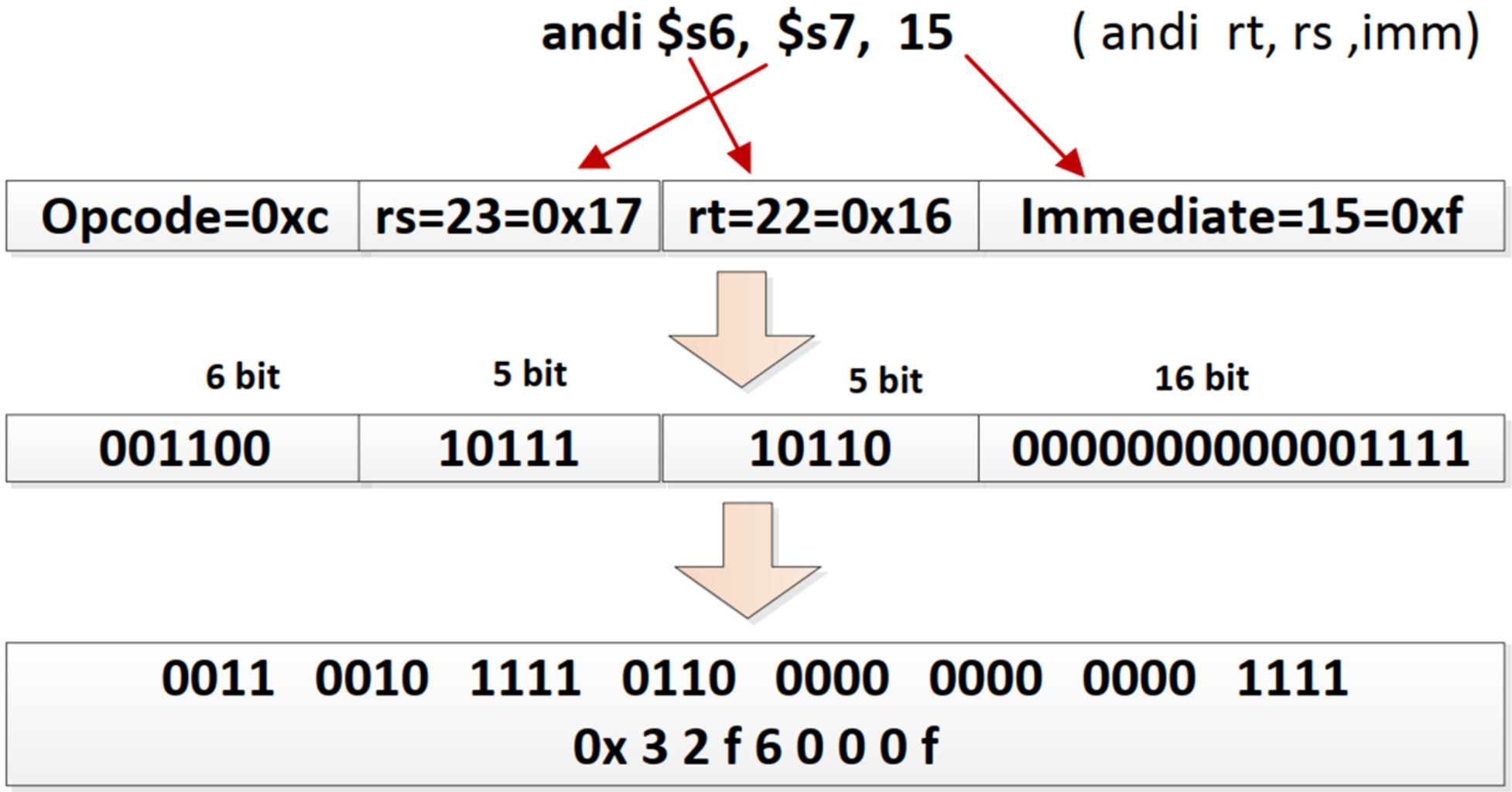
I-Type



- The meaning of each name of I-Type fields
  - op: Basic operation of the instruction
  - rs: Role depends on the instruction
    - Example: in a load word instructions, rs holds the base address
  - rt: Role depends on the instruction
    - Example: in a load word instruction: rt specifies the destination register, which receives the result of the load
  - Constant/Address (Immediate!)
    - Limited to $\pm 2^{15}$
    - There is a special interpretation for conditional jump instructions

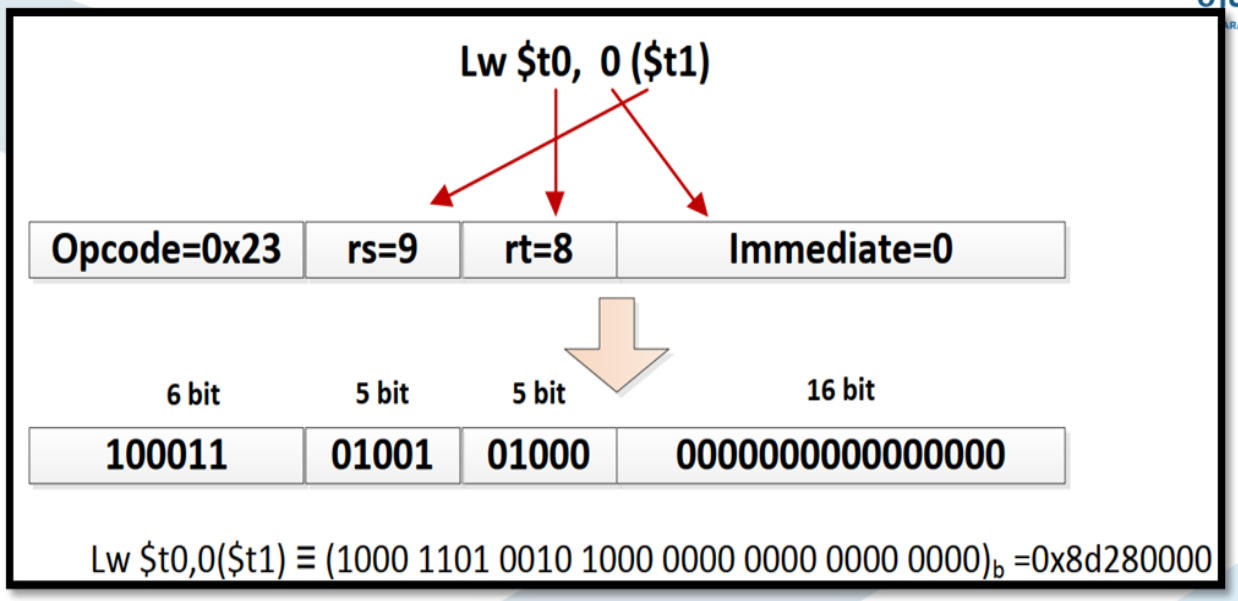
I-Type-Example-1

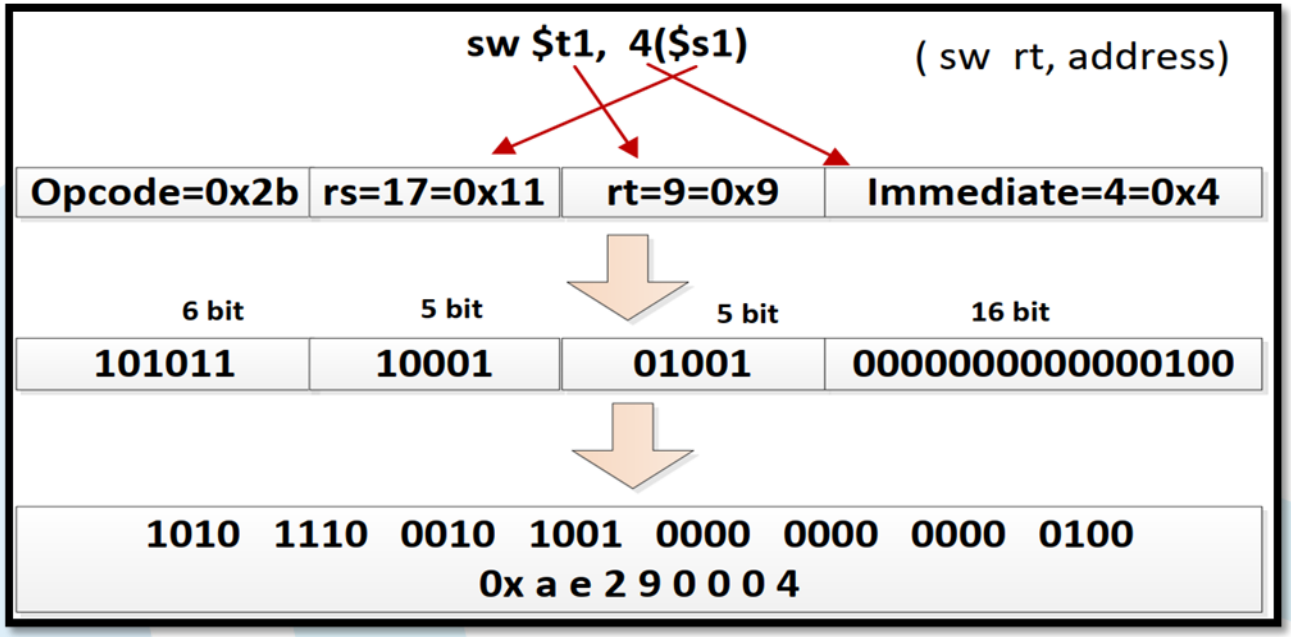
تحويل تعليمات لغة MIPS إلى تعليمات الآلة الموافقة



I-Type-Example-2

تحويل تعليمات لغة MIPS إلى تعليمات الآلة الموافقة





I-Type-Example-3

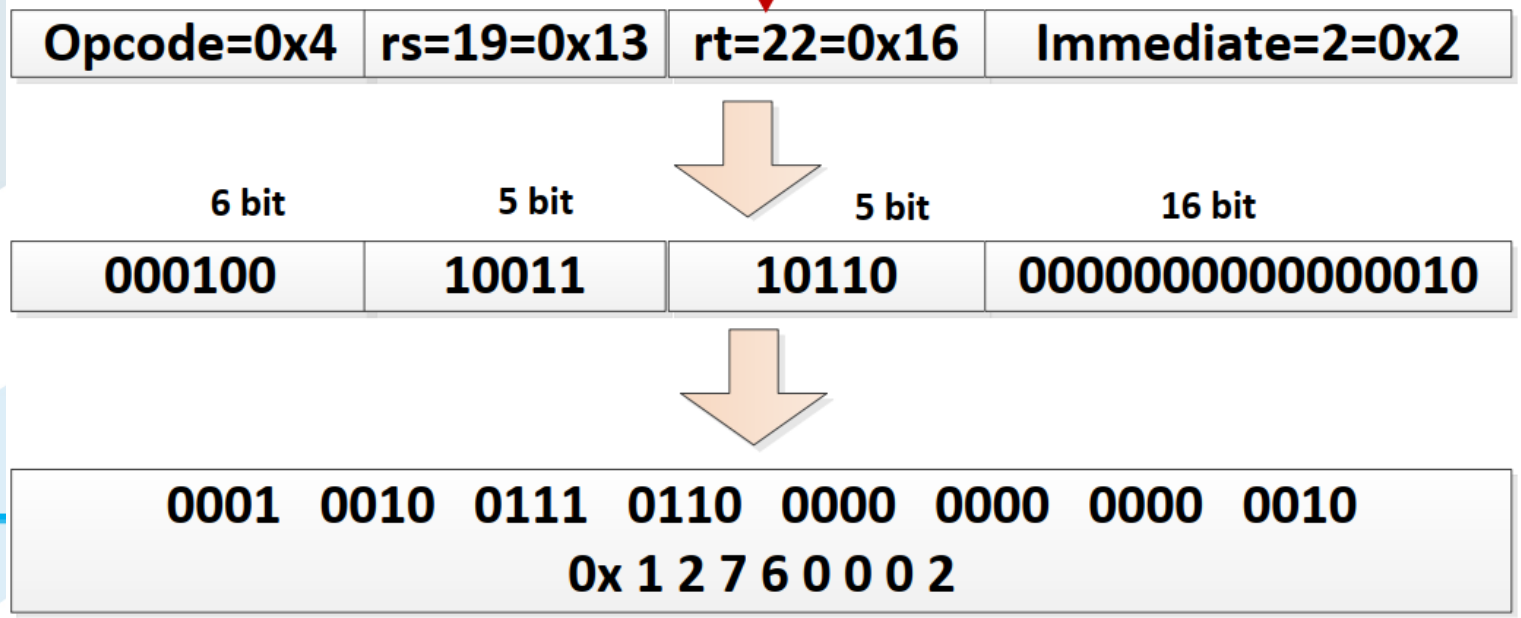
تحويل تعليمات لغة MIPS إلى تعليمات الآلة الموافقة

نريد تحويل التعليمة beq إلى لغة الآلة:

```
    ori $s6,$s6,4
    ori $s3,$s3,3
    beq $s3,$s6, hh
    andi $s6,$s7,15
    sw $t1,4($s1)
hh: add $s0,$s6,$
    ori $s1,$s1,5
```

التعليمات التي سيتم تجاوزها في حال تحقق الشرط = 2

beq $s3, $s6, hh



I-Type-Example-4

تحويل تعليمات لغة MIPS إلى تعليمات الآلة الموافقة

جَـامعة المَـنارة MANARA UNIVERSITY

نريد تحويل التعليمة beq إلى لغة الآلة (لاحظ الفرق مع الحالة السابقة):

```
     ori $s6,$s6,4
hh:
     ori $s3,$s3,3
     andi $s6,$s7,15
     andi $s6,$s6,1
     andi $s0,$s6,2
     lui $s2,0x2222
     beq $s3,$s6, hh
     andi $v0,$0,5
```

التعليمات التي سيتم تجاوزها في حال تحقق الشرط = 4

الخطوات

- نعد التعليمات التي سيتم تجاوزها في حال تحقق الشرط (وهي هنا 4 تعليمات)
- نضيف 2 إلى عدد التعليمات (أي 4+2=6)
- نأخذ المتمم الثنائي للعدد الناتج (نبدل الأصفار بواحدات ثم نجمع 1 للناتج النهائي)

4+2 =6 ≡ 0000 0000 0000 0000 0110

المتمم الثنائي هو :

Immediate = 1111 1111 1111 1111 1001 +1 = 1111 1111 1111 1111 1010 = 0xfffa

لغة الآلة : **0x1276fffa**

I-Type-Example-5

MIPS instruction

```
addi   $21, $22, -50
```

Field representation in decimal:

8	22	21	-50

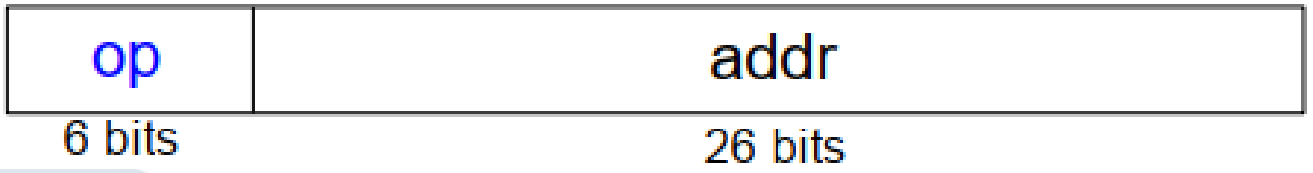
Field representation in binary:

001000	10110	10101	1111111111001110

Hexadecimal representation of instruction:

$$2\ 2\ D\ 5\ F\ F\ C\ E_{16}$$

J-Type



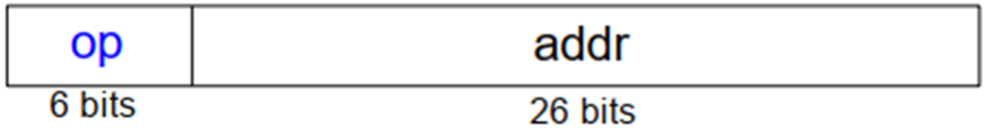
- J-Type is used only for
  - Jump instruction j (opcode = 2)
  - Jump and Link instruction jal (opcode = 3)
- Address field is 26 bit wide
  - Address space = 2^{26}
  - MIPS interprets this space as word address
  - i.e. the address space is 2^{28} byte address
- The upper 4 bits of the program counter are not affected by the jump!

J-Type-Example(1/2)

تحويل تعليمات لغة MIPS إلى تعليمات الآلة الموافقة

نريد تحويل التعليمة j kk إلى لغة الآلة:

```
      or $s6,$t3,$s0
      sll $s3,$s3,3
       j kk
      andi $s6,$s7,15
      sw $t1,4($s1)
kk:   add $s0,$s6,$t0
      ori $s1,$s1,!
```



- قيمة opcode=2 hex
- أما قيمة address المؤلفة من 26bit فتمثل العنوان الذي سيتم القفز إليه (وذلك بعد أن نقوم بتحويل قيمة العنوان إلى قيمة جديدة تمثل ترتيب التعليمة التي سيتم القفز إليها ضمن الكود)
- يتم الحصول على ترتيب التعليمة ضمن الذاكرة من خلال تقسيم عنوانها على 4 (كون كل تعليمة حجمها 4byte)
- ولكن عنوان التعليمة في أغلب الأحيان يكون ذو قيمة ست عشرية (مثلا 0x0040003c) **وبالتالي كيف سنقوم بعملية تقسيم؟؟؟**
- يتم ذلك من خلال إجراء عملية إزاحة للعنوان بمقدار 2 لليمين
- أي نحول العنوان إلى نظام العد الثنائي ثم نجري عملية إزاحة إلى اليمين بمقدار 2 ثم نحول الناتج إلى النظام الست عشري من جديد (عملية الإزاحة إلى اليمين بمقدار 2 تكافئ التقسيم على 4 في نظام العد العشري)

J-Type-Example(2/2)

تحويل تعليمات لغة MIPS إلى تعليمات الآلة الموافقة

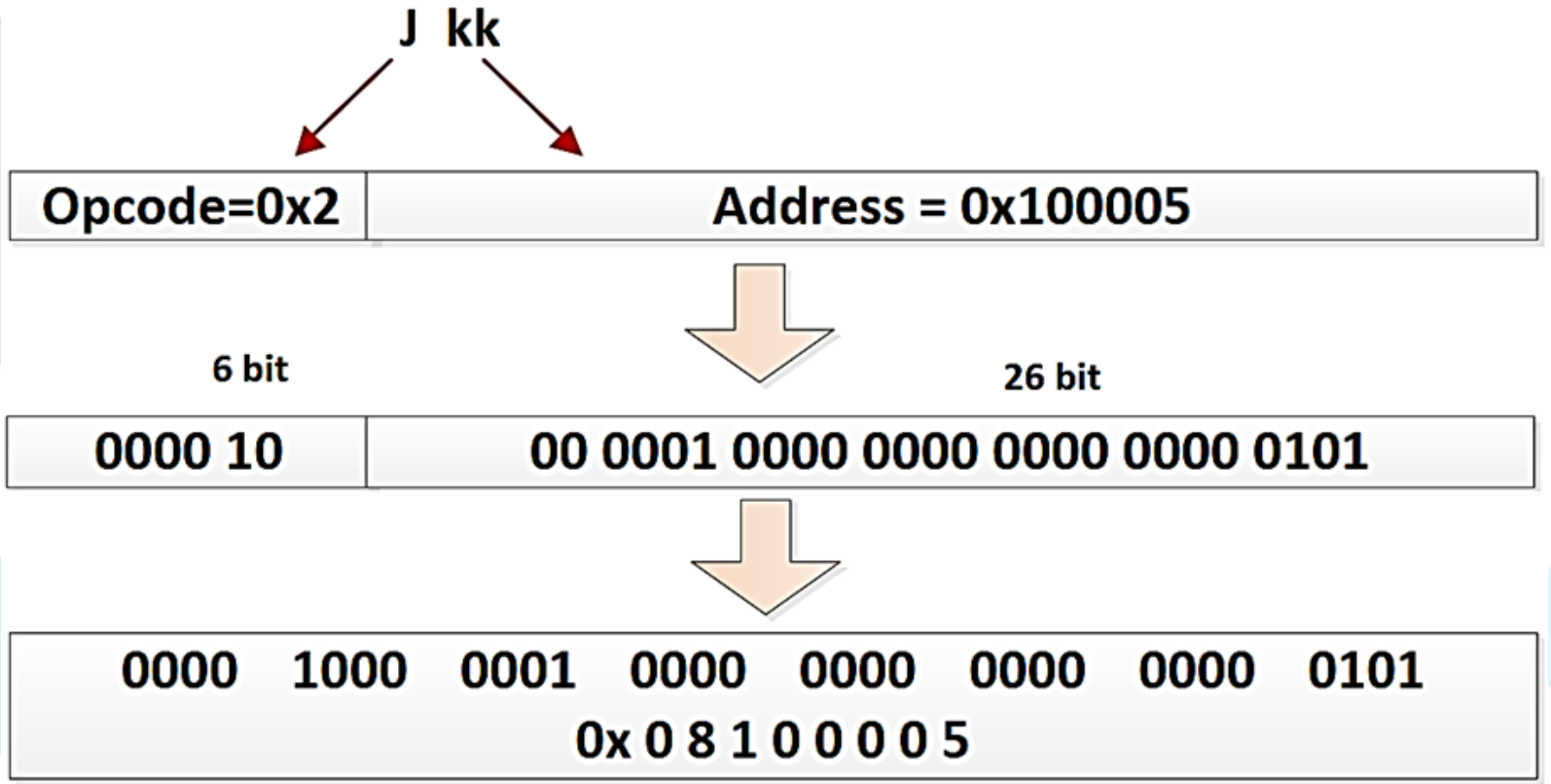
جَـامعة المَـنارة MANARA UNIVERSITY

تم تحديد عناوين التعليمات في ذاكرة التعليمات

```
0x00400000        or $s6,$t3,$s0
0x00400004        sll $s3,$s3,3
0x00400008        j kk
0x0040000c        andi $s6,$s7,15
0x00400010        sw $t1,4($s1)
0x00400014   kk:  add $s0,$s6,$t0
0x00400018        ori $s1,$s1,5
```

لاحظ

ان عنوان الافتة kk هو 0x00400014 وبإجراء عملية ا احة لهذا العنوان نحو اليمين بمقدار 2bit (بعد تحويله إلى الثنائي) يصبح 0x00100005



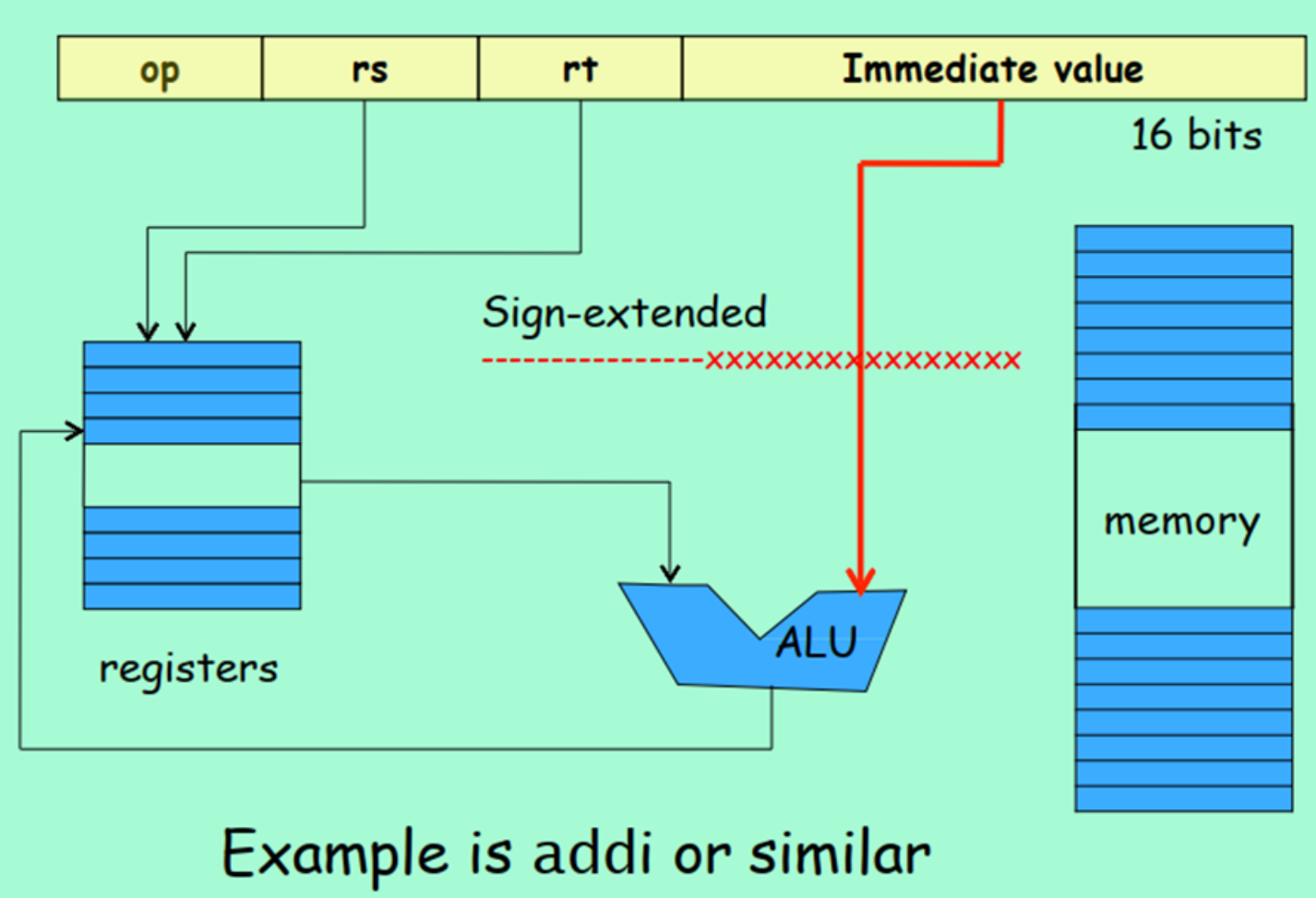
MIPS Addressing Modes(1/8)

- Addressing Mode: how operands are identified
- MIPS supports 5 addressing modes
  - Immediate addressing
  - Register addressing
  - Base or displacement addressing (Data in Memory)
  - PC-relative addressing (Instructions in Memory)
  - Pseudodirect addressing (Instructions in Memory)

أنماط العنونة في MIPS

MIPS Addressing Modes(2/8)

- Immediate addressing
  - The operand is a constant (16 bit) within the instruction itself



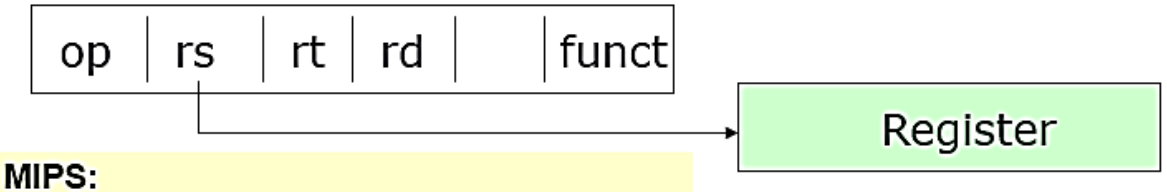
MIPS:
`addi, andi, ori, xori, slti`

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

أنماط العنونة في MIPS

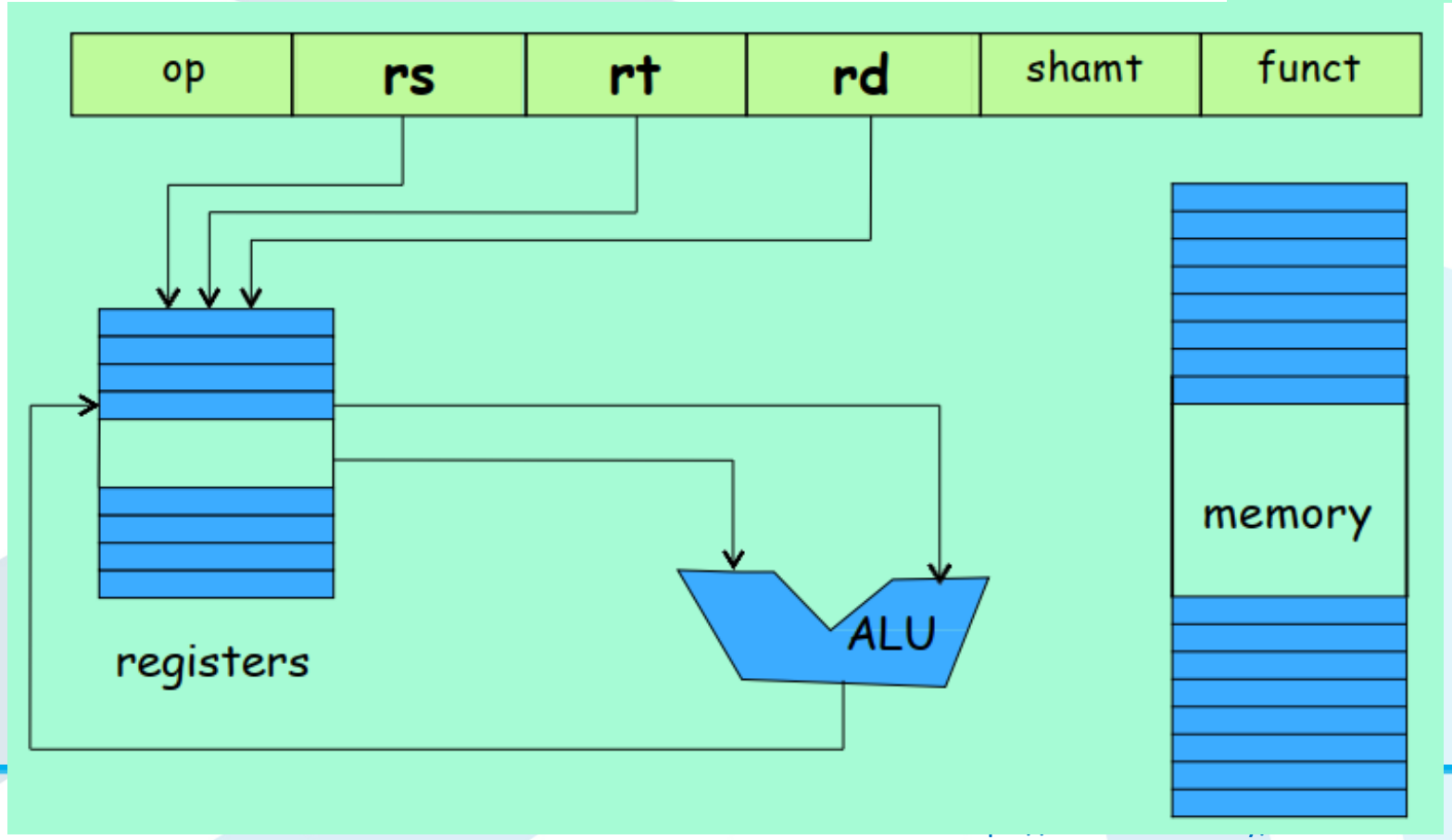
MIPS Addressing Modes(3/8)

- **Register addressing**: operand is a register



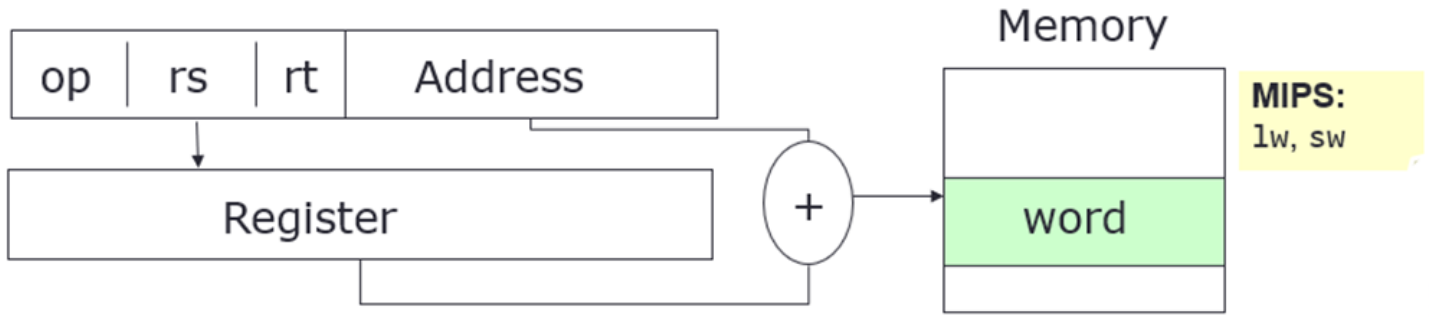
MIPS:
add, sub, and, or, xor, nor, slt, sll, srl

- Example: add $3,$4,$5
- Takes n bits to address 2^n registers



MIPS Addressing Modes(4/8)

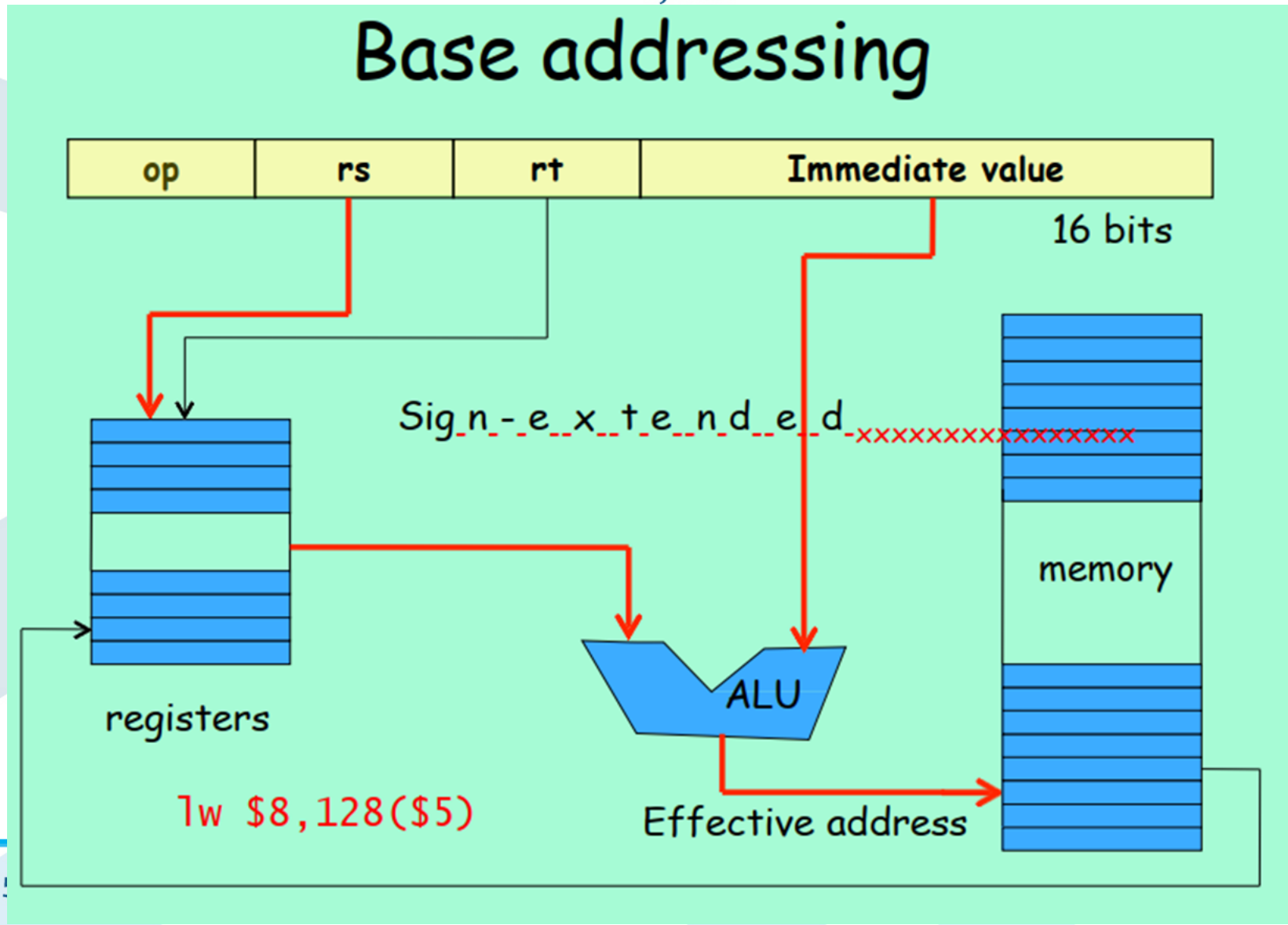
- **Base addressing (displacement addressing)**: operand is at the memory location whose address is sum of a register and a constant in the instruction (`lw`, `sw`)



- 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

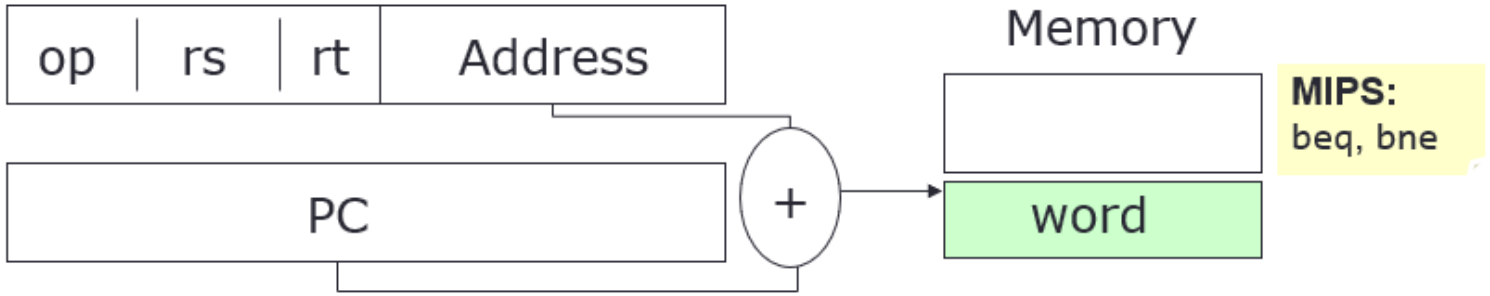
أنماط العنونة في MIPS
MIPS Addressing Modes(5/8)



أنماط العنونة في MIPS

MIPS Addressing Modes(6/8)

- **PC-relative addressing**: address is sum of PC and constant in the instruction (**beq**, **bne**)

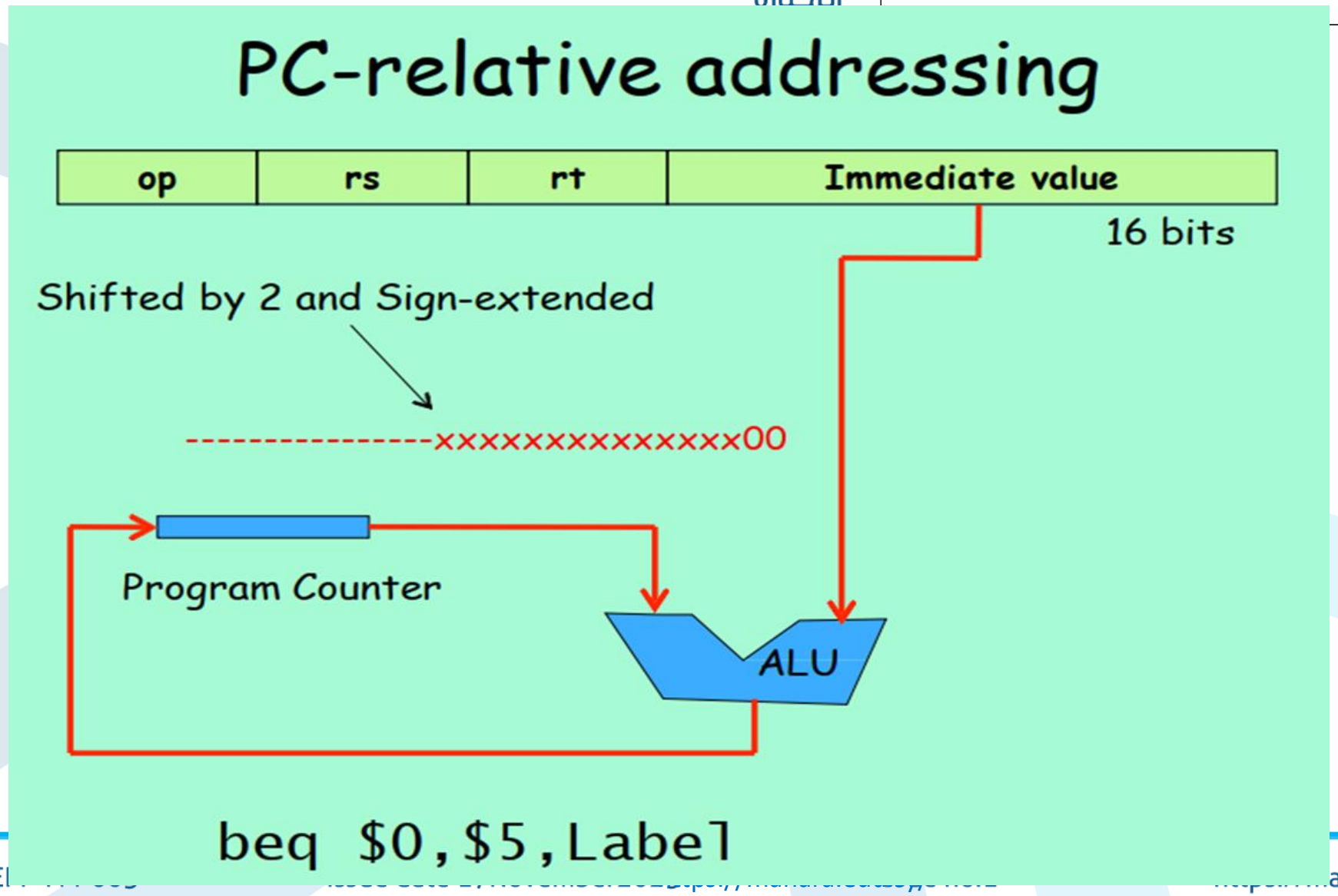


PC-relative addressing: the value in the immediate field is interpreted as an offset of the next instruction (PC+4 of current instruction)

Example: beq $0,$3,Label

op	rs	rt	Immediate value

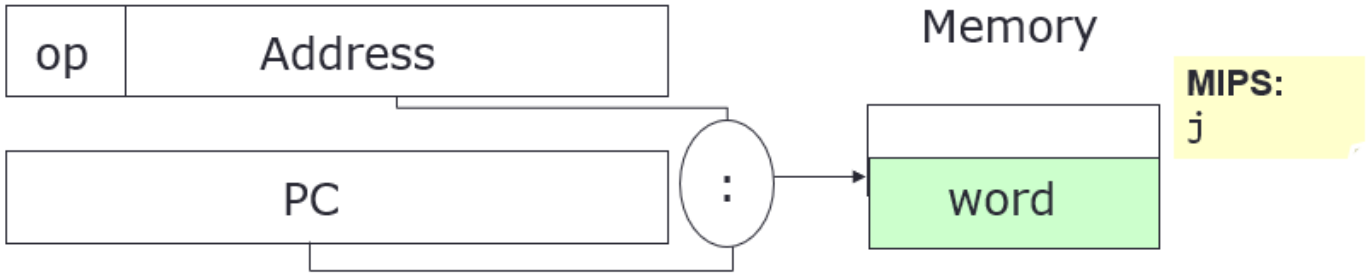
أنماط العنونة في MIPS
MIPS Addressing Modes(7/8)



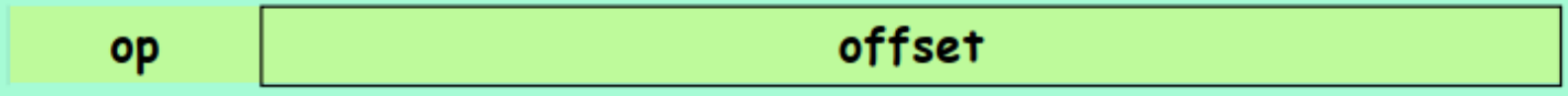
أنماط العنونة في MIPS

MIPS Addressing Modes(8/8)

- **Pseudo-direct addressing**: 26-bit of instruction concatenated with upper 4-bits of PC (`j`)



Example: j Label



PC:	0111	0001 …	… 00
offs:		0101 0001 0100 0010 1111 0101 10	
shift:			00
ADDR:	0111	0101 0001 0100 0010 1111 0101 10	00

جَـامعة
المَـنارة
MANARA UNIVERSITY

نهاية المحاضرة الرابعة