جامعة المنارة الخاصة
كلـــــية الهندســـــة
هندسة ميـــكاترونيك

المعالجات الصغرية ولغة التجميع

المحاضرة الثانية

مدرس المقرر
د. بسام حسن

2025_2026

مفردات من المحاضرة الثانية :

- معالج MIPS
- مسجلات المعالج
- Memory Operands
- الأصناف الأساسية لتعليمات MIPS
- التعليمات الحسابية الأساسية
- التعليمات المنطقية الأساسية
- تعليمات نقل المعطيات بين الذاكرة والمسجلات
- تعليمات القفز المشروط
- تعليمات القفز غير المشروط

معالج MIPS

MIPS اختصار **Microprocessor without Interlocked Pipelines**

أُدخلت معمارية MIPS عام 1985 وامتدت عبر أجيال MIPS I حتى MIPS V ثم إصدارات أحدث مثل MIPS32 و MIPS64

- The MIPS ISA has 32 registers
- Each register is 32-bit wide
- A 32-bit entity (4 bytes) is referred to as a word

- Instructions operate on varying data types
- 8 bits = 1 byte
- 16 bits = 2 bytes = 1 half word
- 32 bits = 4 bytes = 1 word

مسجلات في MIPS32: عددها 32 مسجلاً عرض المسجل: 32 بت = كلمة Word

والمسجلات لها أسماء ولها أرقام موافقة:

جَامعة المَنارة
MANARA UNIVERSITY

Register Operands: MIPS Registers

32 general-purpose registers

#	Name	Usage
0	$zero	The constant value 0
1	$at	Assembler temporary
2	$v0	Values for results and expression evaluation
3	$v1	
4	$a0	Arguments
5	$a1	
6	$a2	
7	$a3	
8	$t0	Temporaries (Caller-save registers)
9	$t1	
10	$t2	
11	$t3	
12	$t4	
13	$t5	
14	$t6	
15	$t7	

#	Name	Usage
16	$s0	Saved temporaries (Callee-save registers)
17	$s1	
18	$s2	
19	$s3	
20	$s4	
21	$s5	
22	$s6	
23	$s7	
24	$t8	More temporaries (Caller-save registers)
25	$t9	
26	$k0	Reserved for OS kernel
27	$k1	
28	$gp	Global pointer
29	$sp	Stack pointer
30	$fp	Frame pointer
31	$ra	Return address

Memory Operands

الذاكرة في MIPS32: حجمها 2^{30} كلمة = 2^{32} بايت و يتعامل المعالج مع الذاكرة بتعليمات "نقل المعطيات" وتتم عنونة الذاكرة عن طريق عنونة البايتات:

- الكلمة رقم 0 عنوانها: 0
- الكلمة رقم 1 عنوانها: 4
- الكلمة رقم 2 عنوانها: 8
- الكلمة رقم m عنوانها: 4*m

كيف توزع الكلمة على أربع بايتات في الذاكرة؟ يعتمد MIPS32 مبدأ Big–Endian.

MSB LSB

11110100	10011000	10110111	00001111

32-bit data

الأصناف الأساسية لتعليمات MIPS

- تعليمات حسابية Arithmetic Instructions.
- تعليمات منطقيةLogical Instructions .
- تعليمات نقل المعطيات Logical Instructions.
- تعليمات القفز المشروطLogical Instructions .
- تعليمات القفز غير المشروط Unconditional Branch.

- تطبق التعليمات على عوامل
- العوامل هي
  - مسجلات
  - مواقع في الذاكرة
  - أعداد ثابتة

التعليمات الحسابية الأساسية

- الجمع Add
  - مثال $s1 = $s2 + $s3

```
add   $s1,$s2,$s3
```

- الطرح Sub
  - مثال $s1 = $s2 – $s3

```
sub   $s1,$s2,$s3
```

- الجمع مع عدد فوري (ثابت)add Immediate
  - مثال $s1 = $s2 + 20

```
addi  $s1,$s2,20
```

التعليمات الحسابية الأساسية: مثال

- Compile the following C statements to MIPS Assembly language

```
f = (g + h) - (i + j);
```

  - The variables f, g, h, i, and j are assigned to the registers $s0, $s1, $s2, $s3, and $s4, respectively.

- Solution:

```
add $t0,$s1,$s2 # register $t0 contains g + h
add $t1,$s3,$s4 # register $t1 contains i + j
sub $s0,$t0,$t1 # f gets $t0 - $t1, which is (g + h)-(i + j)
```

التعليمات المنطقية الأساسية

- خمس تعليمات تنفذ Bit-by-Bit
  - and
  - or
  - nor
  - andi :and immediate
  - ori :or immediate
- إزاحة منطقية إلى اليسار Shift Left Logical sll
- إزاحة منطقية إلى اليمين Shift Right Logical srl

التعليمات المنطقية الأساسية

AND

NOTE:
1 if BOTH a and b are 1. Otherwise 0.

a	b	a AND b
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

OR

NOTE:
0 if BOTH a and b are 0. Otherwise 1.

a	b	a OR b
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

NOR

NOTE:
1 if BOTH a and b are 1. Otherwise 0.

a	b	a NOR b
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

XOR

NOTE:
1 if a is NOT the same as b.

a	b	a XOR b
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

التعليمة and و or

- مثال

```
and $t0,$t1,$t2      # reg $t0 = reg $t1 & reg $t2
```

- إذا كانت قيمة $t2 $0000\ 0000\ 0000\ 0000\ 0000\ 1101\ 1100\ 0000_{two}$
- وقيمة $t1 $0000\ 0000\ 0000\ 0000\ 0011\ 1100\ 0000\ 0000_{two}$
- تصبح قيمة $t0 بعد تنفيذ التعليمة

$0000\ 0000\ 0000\ 0000\ 0000\ 1100\ 0000\ 0000_{two}$

- مثال

```
or $t0,$t1,$t2 # reg $t0 = reg $t1 | reg $t2
```

- إذا كانت قيمة $t2 $0000\ 0000\ 0000\ 0000\ 0000\ 1101\ 1100\ 0000_{two}$
- وقيمة $t1 $0000\ 0000\ 0000\ 0000\ 0011\ 1100\ 0000\ 0000_{two}$
- تصبح قيمة $t0 بعد تنفيذ التعليمة

$0000\ 0000\ 0000\ 0000\ 0011\ 1101\ 1100\ 0000_{two}$

Constants

- Often want to be able to specify operand in the instruction: immediate or literal
- Use the `addi` instruction

```
addi dst, src1, immediate
```

ما هي القيمة الفورية: قيمة ثابتة مكتوبة مباشرة داخل التعليمة بدل قراءتها من الذاكرة أو من مسجل آخر.

- The immediate is a 16 bit signed value between -2^{15} and $2^{15}-1$
- Sign-extended to 32 bits

- Consider the following C code

```
a++;
```

- The `addi` operator

```
addi $s0, $s0, 1          # a = a + 1
```

Shift Left Logical : sll

Shift Right Logical : srl

- sll
  - مثال
```
sll   $t2,$s0,4    # reg $t2 = reg $s0 << 4 bits
```
  - إذا كانت قيمة $s0

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1001_{two} = 9_{ten}$

  - تصبح قيمة $t2 بعد تنفيذ التعليمة

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1001\ 0000_{two} = 144_{ten}$

Shift Left Logical : sll

Shift Right Logical : srl

شرح تعليمة sll :

تعني هذه التعليمة `sll $t1,$s3,2` : إزاحة القيمة الموجودة في $s3 إزاحة منطقية (أي بعد تحويلها إلى الثنائي) نحو اليسار بمقدار 2 bits (خانتين ثنائيتين) ووضع النتيجة في المسجل $t1. فعلى فرض أن قيمة المسجل $s3 =ababcdcd ونريد تنفيذ التعليمة السابقة، نقوم أولاً بتحويل القيمة إلى النظام الثنائي :

`$s3=1010 1011 1010 1011 1100 1101 1100 1101`

نزيح الخانات مرتين إلى اليسار(أي نحذف البتين اليساريين الأعلى أهمية ونضع بدلاً منهما صفرين إلى اليمين) فتصبح قيمة $t1 :

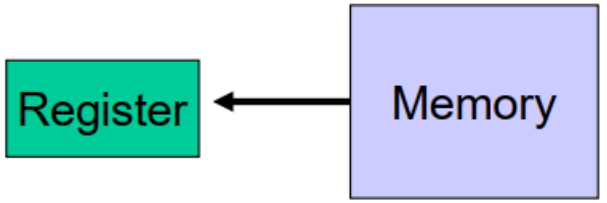
`$t1= 1010 1110 1010 1111 0011 0111 0011 0100 = 0xaeaf3734`

بينما تبقى قيمة $s3= ababcdcd كما هي دن تغيير.

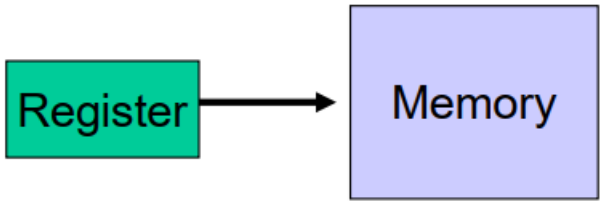
ملاحظة هامة : إن الإزاحة نحو اليسار بمقدار n خانة في النظام الثنائي تكافيء الضرب بـ 2^n في النظام العشري.

تعليمات نقل المعطيات

Load word
lw $t0, memory-address



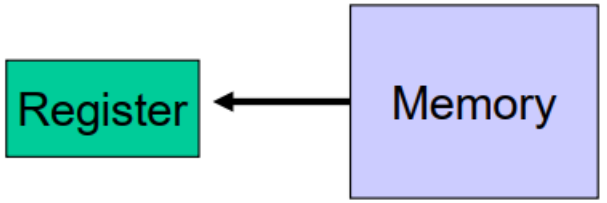
Store word
sw $t0, memory-address



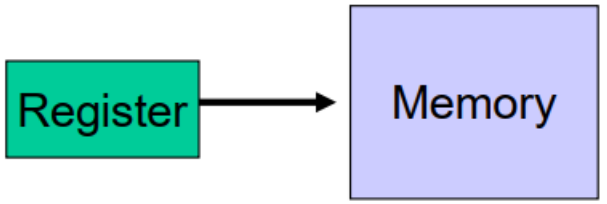
- The format of a load instruction:

تعليمات نقل المعطيات

Load word
lw $t0, memory-address



Store word
sw $t0, memory-address



- The format of a store instruction:

تعليمات نقل المعطيات

- مثال: التعامل مع سلاسل معطيات

Let's assume that A is an array of 100 words and that the compiler has associated the variables g and h with the registers $s1 and $s2 as before. Let's also assume that the starting address, or *base address*, of the array is in $s3. Compile this C assignment statement:

```
g = h + A[8];
```

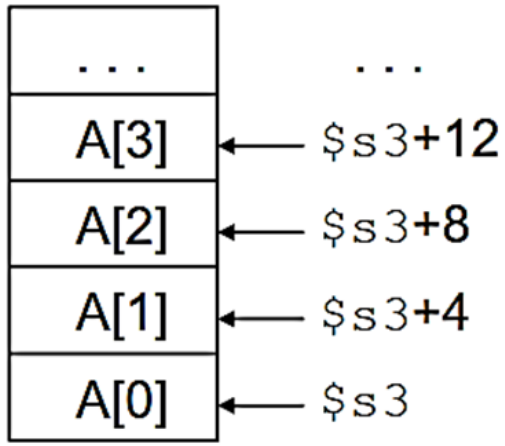
- الحل:

```
lw   $t0,32($s3)  # Temporary reg $t0 gets A[8]
add  $s1,$s2,$t0 # g = h + A[8]
```

تعليمات نقل المعطيات

- Assuming variable b is stored in `$s2` and that the base address of array A is in `$s3`, what is the MIPS assembly code for the C statement

```
A[8] = A[2] - b
```




```
lw   $t0, 8($s3)
sub  $t0, $t0, $s2
sw   $t0, 32($s3)
```

تعليمات نقل المعطيات

load word	lw $s1,20($s2)	$s1 = Memory[$s2 + 20]
store word	sw $s1,20($s2)	Memory[$s2 + 20] = $s1
load half	lh $s1,20($s2)	$s1 = Memory[$s2 + 20]
load half unsigned	lhu $s1,20($s2)	$s1 = Memory[$s2 + 20]
store half	sh $s1,20($s2)	Memory[$s2 + 20] = $s1
load byte	lb $s1,20($s2)	$s1 = Memory[$s2 + 20]
load byte unsigned	lbu $s1,20($s2)	$s1 = Memory[$s2 + 20]
store byte	sb $s1,20($s2)	Memory[$s2 + 20] = $s1
load linked word	ll $s1,20($s2)	$s1 = Memory[$s2 + 20]
store condition. word	sc $s1,20($s2)	Memory[$s2+20]=$s1;$s1=0 or 1
load upper immed.	lui $s1,20	$s1 = 20 * 2^{16}

تعليمات القفز

- تنفذ التعليمات, عادة, وفق تسلسل تخزينها في الذاكرة
  - موافق ببساطة لترتيب كتابتها في البرنامج
  - لمعرفة التعليمة التي يجب تنفيذها يستخدم مسجل Program Counter PC
  - يحوي PC عنوان التعليمة المراد تنفيذها
  - يتم الانتقال إلى التعليمة التالية بتغيير قيمة PC

$$PC = PC + size\ of\ instruction$$

وفقا لبنية MIPS يكون حجم التعليمة في الذاكرة 4 بايت = 32 بت

بالتالي , $PC = PC + 4$

  - إلا إذا حصل قفز!
- القفز هو تغيير في قيمة PC للانتقال إلى عنوان في الذاكرة وتنفيذ التعليمات ابتداء منه

تعليمات القفز

- تعليمات القفز المشروط Conditional Branch
  - قفز عند التساوي Branch on equal beq
  - قفز عند عدم التساوي branch on not equal
  - تنصيب في حالة أصغر set on less than slt
  - تنصيب في حالة أصغر, بدون إشارة set on less than unsigned sltu
  - تنصيب في حالة أصغر مع قيمة مباشرة set on less than immediate slti
  - تنصيب في حالة أصغر مع قيمة مباشرة, بدون إشارة set on less than immediate unsigned sltui
- تعليمات القفز غير المشروط Unconditional Branch
  - قفز jump j
  - قفز إلى عنوان موجود في مسجل jump register jr
  - قفز مع ارتباط jump and link jal

تعليمات القفز المشروط

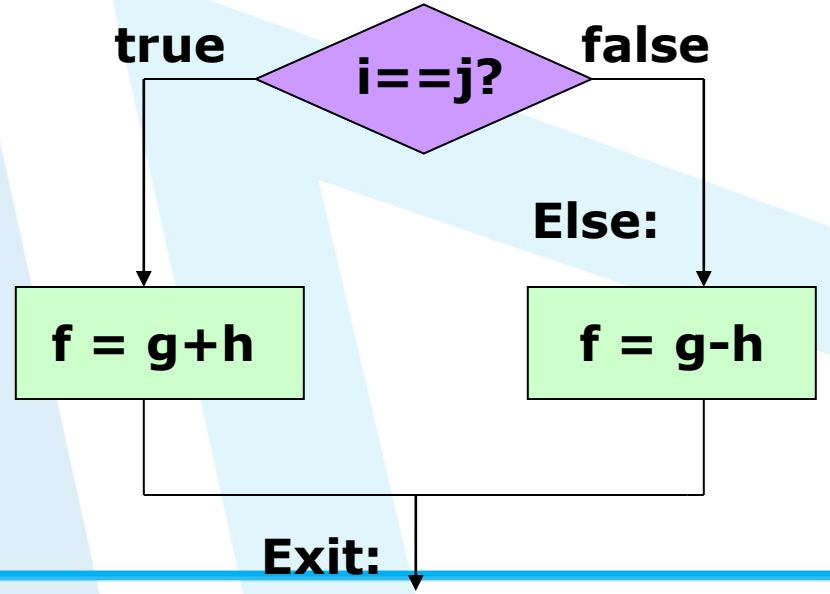
- تقوم باختبار تساوي معاملين وإجراء قفز إلى عنوان محدد (Go to) عند تحقق المساواة/عدم المساواة
- تعليمة القفز عند التساوي Branch if equal
  - الصيغة `beq register1, register2, L1`
  - يتم القفز إلى اللافتة L1 عند تساوي القيمة في المسجلين
- تعليمة القفز عند عدم التساوي Branch if not equal
  - الصيغة `bne register1, register2, L1`
  - يتم القفز إلى اللافتة L1 عند اختلاف قيمة المسجلين

تعليمات القفز المشروط: مثال if-else

C Statement to translate	Variables Mapping

`if (i == j)`
`  f = g + h;`
`else`
`  f = g - h;`
	f ➔ $s0
	g ➔ $s1
	h ➔ $s2
	i ➔ $s3
	j ➔ $s4

```
      bne $s3, $s4, Else
      add $s0, $s1, $s2
      j   Exit
Else: sub $s0, $s1, $s2
Exit:
```



Question: Rewrite with **beq**?

Compiling a *while* Loop in C

Here is a traditional loop in C:

```
while (save[i] == k)
        i += 1;
```

Assume that `i` and `k` correspond to registers `$s3` and `$s5` and the base of the array `save` is in `$s6`. What is the MIPS assembly code corresponding to this C segment?

```
Loop: sll  $t1,$s3,2     # Temp reg $t1 = i * 4
      add  $t1,$t1,$s6   # $t1 = address of save[i]
      lw   $t0,0($t1)    # Temp reg $t0 = save[i]
      bne  $t0,$s5, Exit # go to Exit if save[i] ≠ k
      addi  $s3,$s3,1    # i = i + 1
      j    Loop          # go to Loop
Exit:
```

Exercise : Simple Loop

- Given the following MIPS code:

```
        add  $t0, $zero, $zero
        add  $t1, $t0, $t0
        addi $t2, $t1, 4
Again:  add  $t1, $t1, $t0
        addi $t0, $t0, 1
        bne  $t2, $t0, Again
```

i. How many instructions are executed? **Answer: (c)**

(a) 6 (b) 12 (c) 15 (d) 18 (e) None of the above

ii. What is the final value in `$t1`? **Answer: (c)**

(a) 0 (b) 4 (c) 6 (d) 10 (e) None of the above

تعليمات المقارنة

- تنصيب في حالة أصغر set on less than slt
  - slt $t0, $s3, $s4 # $t0 = 1 if $s3 < $s4
- تنصيب في حالة أصغر, بدون إشارة set on less than unsigned sltu
- تنصيب في حالة أصغر مع قيمة مباشرة set on less than immediate slti
  - slti $t0,$s2,10 # $t0 = 1 if $s2 < 10
- تنصيب في حالة أصغر مع قيمة مباشرة, بدون إشارة set on less than immediate unsigned sltui
- ملاحظة
  - من تعليمات القفز المشروط (وتعليمات المقارنة) يقوم المترجم بتحقيق كل حالات القفز المشروط الأخرى
    - قفز في حالة أصغر أو يساوي
    - قفز في حالة أكبر
    - قفز في حالة أكبر أو يساوي
  - أي يتم تحقيق حالات القفز هذه باستخدام تعليمتين بسيطتين, بدلا من تعليمة واحدة معقدة

تعليمات القفز غير المشروط

- قفز jump j
  - قفز غير مشروط إلى عنوان محدد بقيمته (أو بلافتة)
  - مثال j label1
  - مثال j 10000
- قفز إلى عنوان موجود في مسجل jump register jr
  - قفز غير مشروط إلى عنوان محدد بمسجل
  - تستخدم عادة مع المسجل $ra للعودة من البرامج الفرعية

```
jr $ra
```

- قفز مع ارتباط jump and link jal
  - قفز إلى مكان محدد مع وضع عنوان التعليمة التالية لها في المسجل $ra
    - يخزن في $ra القيمة PC + 4
    - عنوان التعليمة التالية لها هو "الارتباط"
  - تستخدم للقفز إلى البرامج الفرعية Procedures ويسمى الارتباط عندها بعنوان العودة Return Address

جَـامعة
المَـنارة
MANARA UNIVERSITY

نهاية المحاضرة الثانية