كلية الهندسة
قسم هندسة الميكاترونيك

جَامعة المَنارة
MANARA UNIVERSITY

بنيان الحواسيب

محاضرة 4

Instructions: Language of the Computer

د. فادي متوج

- لإعطاء أوامر لجهاز الكمبيوتر، يجب أن نتحدث لغته. تسمى الكلمات المكونة للغة الكمبيوتر **بالتعليمات (Instructions)**، وتسمى مفرداتها مجموعة التعليمات(Instruction Set).
- ربما يظن البعض أن لغات أجهزة الكمبيوتر متنوعة مثل تلك الخاصة بالناس، ولكن في الواقع، فإن لغات الكمبيوتر متشابهة تمامًا . يحدث هذا التشابه لأن جميع أجهزة الكمبيوتر مصنعة من تقنيات تعتمد على مبادئ أساسية مماثلة ولأن هناك بعض العمليات الأساسية التي يجب أن توفرها جميع أجهزة الكمبيوتر.
- في هذه المحاضرة، سندرس مجموعة التعليمات الخاصة بجهاز كمبيوتر حقيقي، كصيغة مكتوبة من قبل الأشخاص و كصيغة مقروءة من قبل الكمبيوتر.

التعليمات الحسابية

مثال: إن مهمة المترجم (Compiler) هي إسناد متغيرات البرنامج إلى المسجلات. على سبيل المثال في العلاقة:

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

تم إسناد المتغيرات f و g و h و i و j للمسجلات $s0 و $s1 و $s2 و $s3 و $s4 على الترتيب. ما هو كود MIPS الناتج عن المترجم؟

الحل:

```
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)
```

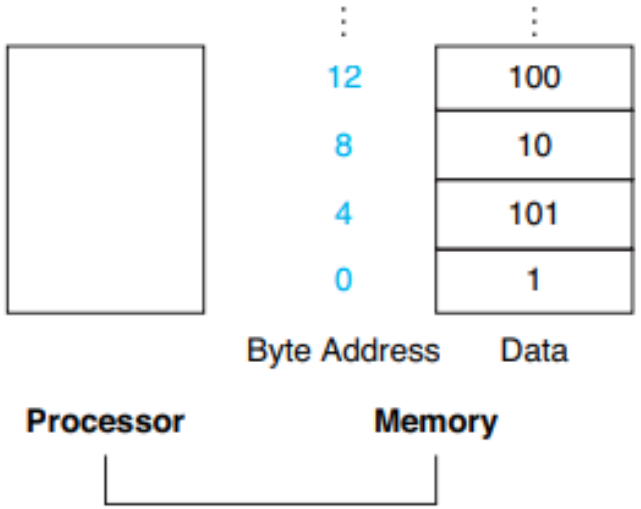
عناوين الذاكرة ومحتويات الذاكرة في تلك المواقع

- تحتوي العديد من البرامج على متغيرات أكثر من عدد المسجلات التي تمتلكها أجهزة الكمبيوتر. وبالتالي، يحاول المترجم الاحتفاظ بالمتغيرات الأكثر استخدامًا في المسجلات ويضع الباقي في الذاكرة، باستخدام تعليمات النقل لنقل المتغيرات بين المسجلات والذاكرة.
- عادةً ما يُطلق على تعليمة نقل البيانات التي تنسخ البيانات من الذاكرة إلى المسجل تعليمة **load** .
- صيغة تعليمة **load** هي كما يلي: اسم العملية متبوعًا بالمسجل المراد تحميله بالقيمة المسحوبة من الذاكرة، ثم قيمة ثابتة ومسجل مستخدم للوصول إلى الذاكرة.
- يشكل مجموع القيمة الثابتة من التعليمة ومحتويات المسجل الثاني عنوان موقع الذاكرة المطلوب الوصول إليه.
- الاسم الفعلي لهذه التعليمة في قائمة تعليمات MIPS هو **lw**، وهي اختصار لـ load word

عناوين الذاكرة ومحتويات الذاكرة في تلك المواقع

- تحتوي لغات البرمجة على متغيرات بسيطة تحتوي على عناصر بيانات مفردة، كما هو الحال في المثال السابق ، ولكنها تحتوي أيضًا على بنى بيانات أكثر تعقيدًا مثل المصفوفات.
- يمكن أن تحتوي بنى البيانات المعقدة هذه على العديد من عناصر البيانات أكثر من عدد المسجلات الموجودة في الكمبيوتر. كيف يمكن للكمبيوتر أن يمثل ويصل إلى مثل هذه البنى الكبيرة؟
- يمكن للمعالج الاحتفاظ بكمية صغيرة فقط من البيانات في المسجلات، لكن ذاكرة الكمبيوتر تحتوي على مليارات من عناصر البيانات. وبالتالي، يتم الاحتفاظ ببنى البيانات (المصفوفات) في الذاكرة.
- تحدث العمليات الحسابية فقط على المسجلات في تعليمات MIPS، وبالتالي، يجب أن تتضمن مجموعة تعليمات MIPS تعليمات تنقل البيانات بين الذاكرة والمسجلات. تسمى هذه التعليمات تعليمات نقل البيانات Data Transfer Instructions .
- تستخدم مجموعة تعليمات MIPS في الواقع عنونة البايت،حيث تمثل كل كلمة أربعة بايتات.
- للوصول إلى كلمة في الذاكرة، يجب أن تحتوي التعليمات على عنوان الذاكرة لهذه الكلمة.

عناوين الذاكرة ومحتويات الذاكرة في تلك المواقع



- بالإضافة إلى ربط المتغيرات بالمسجلات، يحجز المترجم لبنى البيانات مثل المصفوفات مواقع في الذاكرة.
- يمكن للمجمع بعد ذلك وضع عنوان البداية الصحيح في تعليمات نقل البيانات.
- نظرًا لأن معظم بنى المعطيات تتعامل مع وحدات البايت الفردية. لذلك، يتطابق عنوان الكلمة مع عنوان واحد من 4 بايت داخل الكلمة، وتختلف عناوين الكلمات المتسلسلة بمقدار 4.
- على سبيل المثال، يوضح الشكل جانباً عناوين MIPS الفعلية للكلمات؛ عنوان البايت للكلمة الثالثة هو 8.
- في MIPS، يجب أن تبدأ الكلمات من العناوين التي تعد من مضاعفات الرقم 4. وهذا المتطلب يسمى تقييد المحاذاة (alignment)، (سنرى لاحقاً أن المحاذاة تؤدي إلى عمليات نقل أسرع للبيانات.)
- تنقسم أجهزة الكمبيوتر إلى تلك التي تستخدم عنوان أقصى اليسار أو بايت "النهاية الكبيرة big end" كعنوان للكلمة مقابل تلك التي تستخدم بايت أقصى اليمين أو "النهاية الصغيرة little end".
- تتبع مجموعة تعليمات MIPS نمط العنونة big end

ترجمة عبارة حسابية عندما تكون المعاملات في الذاكرة

مثال: افترض أن المتغير h مرتبط بالسجل `$s2` وأن العنوان الأساسي للمصفوفة A موجود في `$s2` ما هو برنامج لغة الأسمبلي MIPS للعبارة التالية؟

A[12] = h + A[8];

الحل:

```
lw  $t0,32($s3)   # Temporary reg $t0 gets A[8]
add $t0,$s2,$t0   # Temporary reg $t0 gets h + A[8]
sw  $t0,48($s3)   # Stores h + A[8] back into A[12]
```

معاملات ثابتة أو فورية

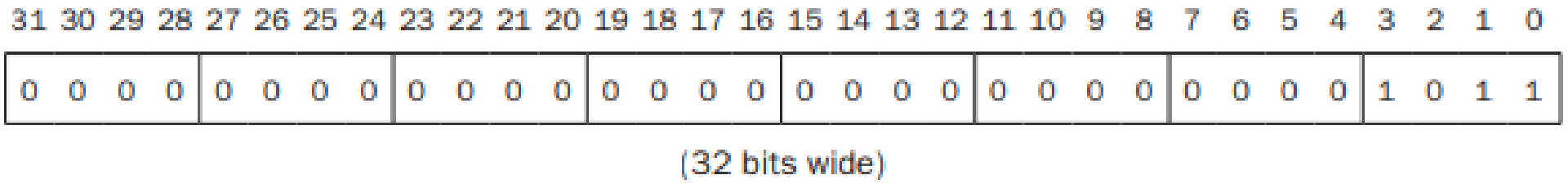
- تقدم مجموعة تعليمات MIPS نسخًا من التعليمات الحسابية يكون فيها أحد المعاملات قيمة ثابتة. يُطلق على تعليمة الجمع مع معامل واحد ثابت اسم add immediate أو **addi** .
- مثال : لإضافة 4 للمسجل $s3 نكتب فقط التعليمة :

```
addi $s3,$s3,4      # $s3 = $s3 + 4
```

- نصادف المعاملات الثابتة بشكل متكرر، وبتضمين الثوابت داخل التعليمات الحسابية، تكون العمليات أسرع بكثير وتستخدم طاقة أقل مما لو تم تحميل الثوابت من الذاكرة.

Signed and Unsigned Numbers

- يتم الاحتفاظ بالأرقام في أجهزة الكمبيوتر كسلسلة من الإشارات الإلكترونية العالية والمنخفضة، وبالتالي يتم تمثيل الأرقام في الحواسيب بالنظام الثنائي.



- يبلغ طول كلمة MIPS 32بتًا، لذا يمكننا تمثيل 2^{32} نمط 32 بت مختلفًا تمثل الأرقام من 0 إلى $(2^{32}-1)$ (أي من 0 إلى 4،294،967،295):

$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 0_{ten}$$
$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = 1_{ten}$$
$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = 2_{ten}$$
$$\ldots \qquad\qquad \ldots$$
$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = 4,294,967,293_{ten}$$
$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = 4,294,967,294_{ten}$$
$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = 4,294,967,295_{ten}$$

Signed and Unsigned Numbers

- برامج الكمبيوتر تتعامل مع كل من الأعداد الموجبة والسالبة، لذلك نحتاج إلى تمثيل نستطيع من خلاله تمييز الأعداد الموجبة عن السالبة.
- في لغة البرمجة C، على سبيل المثال، تسمى الأعداد بإشارة بالأعداد الصحيحة (يتم التصريح عنها كـ int في البرنامج) بينما يتم التصريح عن الأعداد بدون إشارة كـ unsigned int

الأعداد الصحيحة بإشارة
Signed Integers

توجد عدة طرق لتمثيل الأرقام بإشارة (الأرقام الموجبة و السالبة) :

- المتمم الأحادي 1's complement
- المتمم الثنائي 2's complement

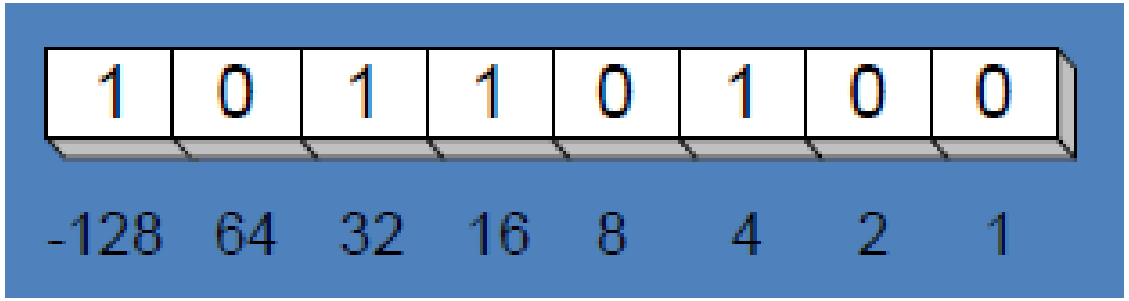
- في الأعداد بإشارة ينقسم المجال إلى جزأين متساويين:
  - الجزء الأول يمثل الأعداد الموجبة
  - الجزء الثاني يمثل الأعداد السالبة
- سيكون التركيز على تمثيل الأعداد بإشارة من خلال المتمم الثنائي باعتبار أنه هو المستخدم على نطاق واسع في الكومبيوتر لتمثيل الأعداد الصحيحة بإشارة .

تمثيل الأعداد بإشارة باستخدام المتمم الثنائي

8-bit Binary value	Unsigned value	Signed value
00000000	0	0
00000001	1	+1
00000010	2	+2
. . .	. . .	. . .
01111110	126	+126
01111111	127	+127
10000000	128	-128
10000001	129	-127
. . .	. . .	. . .
11111110	254	-2
11111111	255	-1

- لإيجاد مطال القيمة بإشارة ممثلة باستخدام المتمم

الثنائي نقوم بإسناد وزن سلبي للبت الأكثر أهمية MSB



$-128 + 32 + 16 + 4 = -76$

مجال الأعداد بإشارة

- من أجل الأعداد بـ n بت يكون المجال :

$$-2^{n-1} \text{ to } (2^{n-1} - 1)$$

Byte	–128 to +127	-2^7 to $(2^7 - 1)$
Half Word	–32,768 to +32,767	-2^{15} to $(2^{15} - 1)$
Word	–2,147,483,648 to +2,147,483,647	-2^{31} to $(2^{31} - 1)$
Double Word	–9,223,372,036,854,775,808 to +9,223,372,036,854,775,807	-2^{63} to $(2^{63} - 1)$

تشكيل المتمم الثنائي

- مثال: مثل العدد (-36)بحسب المتمم الثنائي؟

00100100 = +36	للحصول على العدد -36 ننطلق في البداية من العدد +36
11011011	الخطوة1: نعكس الخانات (إيجاد المتمم الأحادي)
+ 1	الخطوة 2: نجمع 1 إلى القيمة الناتجة من الخطوة1
11011100 = -36	حاصل الجمع = التمثيل بحسب المتمم الثنائي

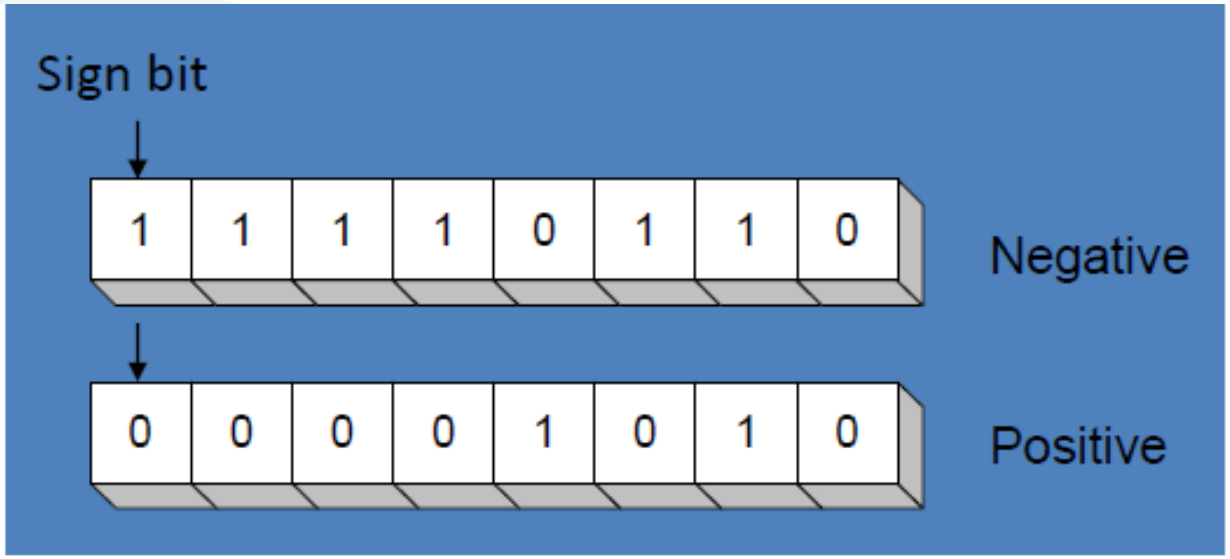
- توجد طريقة أخرى للحصول على المتمم الثنائي:
  - ✓نبدأ بالبحث من الخانة الأقل أهمية عن أول ورود لـ 1
  - ✓نترك كل الأصفار على يمينه بدون تغيير
  - ✓نقلب كل الخانات على يساره (نجعل الأصفار واحدات و بالعكس)

```
Binary Value
= 00100100
2's Complement
= 11011100
```

خانة الإشارة
Sign Bit

1 = negative

0 = positive



توسعة خانة الإشارة

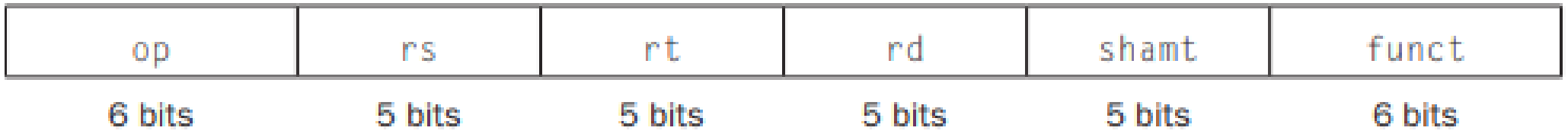
- يمكن توسعة العدد بإشارة من خلال تكرار خانة الإشارة في أقصى اليسار بعدد خانات التوسعة المطلوبة دون أن يؤثر ذلك على إشارة و مطال العددج
- مثال: وسع العدد 10110011 بحيث يصبح طوله 16 بت؟

10110011 = -77 ⟹ 11111111 10110011 = -77

- مثال : وسع العدد 01100010 بحيث يصبح طوله 16 بت؟

01100010 = +98 ⟹ 00000000 01100010 = +98

ترجمة تعليمات لغة التجميع إلى تعليمات لغة الآلة



فيما يلي معنى كل اسم من الحقول في تعليمات MIPS :

- **op** : العملية الأساسية التي تقوم بها التعليمة (حسابية – تحتوي معامل ثابت- تعليمة قفز)، يسمى هذا الحقل رمز العملية opcode.
- **rs** : المسجل الذي يحتوي على المعامل الأول للتعليمة.
- **rt**: المسجل الذي يحتوي على المعامل الثاني للتعليمة.
- **rd**: مسجل الوجهة الذي يخزن نتيجة العملية.
- **shamt**: مقدار الإزاحة. يتم استخدام هذا الحقل مع تعليمات الإزاحة، وبالتالي يحتوي الحقل القيمة صفر من أجل التعليمات الحسابية.
- **funct**: الوظيفة. من خلال هذا الحقل وحقل op السابق يتم تحديد نوع التعليمة الواجب تنفيذها بشكل دقيق

ترجمة تعليمات لغة التجميع إلى تعليمات لغة الآلة

مثال: حول التعليمة التالية من لغة التجميع (لغة الأسمبلي) إلى لغة الآلة `add $t0,$s1,$s2`

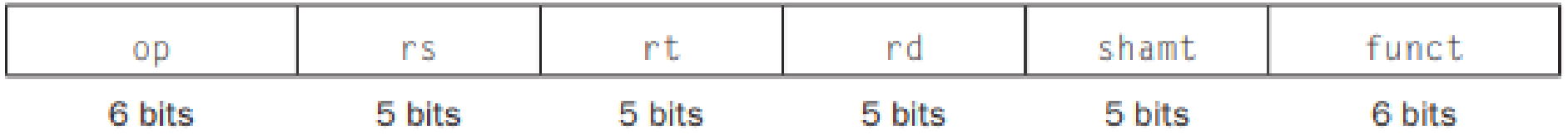
الحل:

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
000000	10001	10010	01000	00000	100000
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

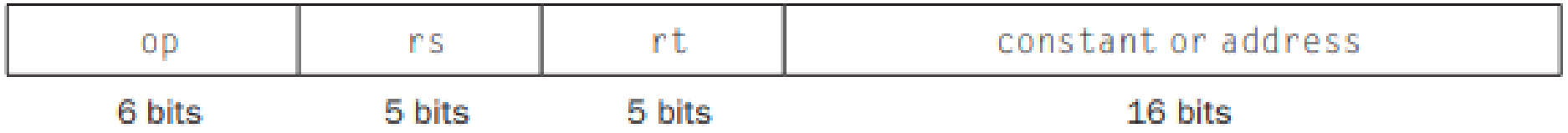
- تقوم هذه التعليمة بجمع محتوى المسجل `$s1` مع محتوى المسجل `$s2` و تضع نتيجة المجموع في المسجل `$t1`
- كل جزء من هذه الأجزاء المكونة للتعليمة يسمى حقل :
  - يخبر الحقلان الأول والأخير (الحقل op و الحقل funct) معًا جهاز الكمبيوتر أن هذه التعليمة تنجز عملية **جمع**.
  - يعطي الحقل الثاني رقم المسجل الذي يمثل المعامل الأول لعملية الجمع (`$s1` = 17)
  - يعطي الحقل الثالث معامل المصدر الآخر لعملية الجمع (`$s2` = 18)
  - يحتوي الحقل الرابع على رقم المسجل الذي سيحصل على المجموع (`$t0` = 8)
  - الحقل الخامس غير مستخدم في هذه التعليمة، لذلك يتم تعيينه على قيمة 0.

صيغ التعليمات
Instruction Formats

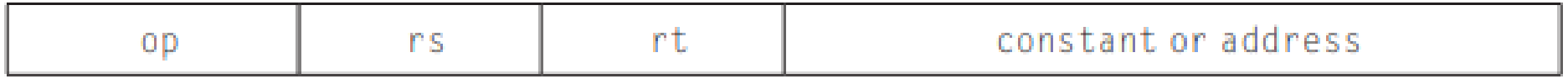
- في مجموعة تعليمات MIPS تمتلك جميع التعليمات نفس الطول (32 بت)
- R-format أو R-type (ترمز R إلى register)



- I-format أو I-type



صيغ التعليمات
Instruction Formats



- I-format أو I-type

```
lw $t0,32($s3) # reg $t0 gets A[8]
```

- يتم وضع 19 (من أجل $s3) في حقل **rs**، و 8 (من أجل $t0) في الحقل **rt**، و 32 في حقل **العنوان**.
- نلاحظ أن معنى الحقل rt قد تغير بالنسبة لهذه التعليمات: في تعليمة lw، يحدد الحقل rt مسجل الوجهة الذي يتلقى نتيجة التعليمة.
- الحقول الثلاثة الأولى من صيغ التعليمات من النوع R و النوع I هي بنفس الحجم ولها نفس الأسماء ؛ طول الحقل الرابع في النوع I يساوي مجموع أطوال الحقول الثلاثة الأخيرة من النوع R.
- يتم تمييز التنسيقات بالقيم الموجودة في الحقل الأول: يتم تعيين مجموعة مميزة من القيم لكل صيغة في الحقل الأول (op) حتى يعرف الجهاز ما إذا كان سيعامل النصف الأخير من التعليمات على أنه ثلاثة حقول (R-type) أو كحقل فردي (I-type).

صيغ التعليمات
Instruction Formats

Instruction	Format	op	rs	rt	rd	shamt	funct	address
add	R	0	reg	reg	reg	0	32_{ten}	n.a.
sub (subtract)	R	0	reg	reg	reg	0	34_{ten}	n.a.
add immediate	I	8_{ten}	reg	reg	n.a.	n.a.	n.a.	constant
lw (load word)	I	35_{ten}	reg	reg	n.a.	n.a.	n.a.	address
sw (store word)	I	43_{ten}	reg	reg	n.a.	n.a.	n.a.	address

مثال

- إذا كان المسجل $t1 يحتوي على العنوان الأساس للمصفوفة A و $s2 يتوافق مع h، فإن التعليمة التالية

```
A[300] = h + A[300];
```

- يتم التعبير عنها بلغة التجميع كما يلي:

```
lw $t0,1200($t1) #reg $t0 gets A[300]

add $t0,$s2,$t0  #reg $t0 gets h + A[300]

sw $t0,1200($t1) # Stores h + A[300] back into A[300]
```

- ما هو كود لغة الآلة MIPS لهذه التعليمات الثلاثة؟

مثال

- الحل:

op	rs	rt	rd	address/ shamt	funct
35	9	8	1200		
0	18	8	8	0	32
43	9	8	1200		

100011	01001	01000	0000 0100 1011 0000		
000000	10010	01000	01000	00000	100000
101011	01001	01000	0000 0100 1011 0000		

Name	Format	Example						Comments
add	R	0	18	19	17	0	32	add $s1,$s2,$s3
sub	R	0	18	19	17	0	34	sub $s1,$s2,$s3
addi	I	8	18	17	100			addi $s1,$s2,100
lw	I	35	18	17	100			lw $s1,100($s2)
sw	I	43	18	17	100			sw $s1,100($s2)
Field size		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions are 32 bits long
R-format	R	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	I	op	rs	rt	address			Data transfer format