المحاضرة 5 عملي معالجات صغرية – ميكاترونيكس

المثال 1: لديك البرنامج المكتوب بلغة التجميع (Assembly) التالي:

البرنامج	الشرح

`.data`
`.word 31, 8, 12, -4, 67, 5, 15`
`.text`
`lui $s0, 0x1001`
`lw $t1, 0($s0)`
`Start:`
`sll $s2, $s6, 2`
`add $s3, $s0 ,$s2`
`lw $s4, 0($s3)`
`beq $s4,$0, exit`
`slt $t0, $s4, $t1`
`bne $t0, $0, next1`
`J next2`
`next1:`
`or $t1, $0, $s4`
`next2: addi $s6, $s6, 1`
`J Start`
`exit:`
`or $s5, $0, $t1`
	إزاحة منطقية للقيمة الموجودة في المسجل $s6 بمقدار خانتين نحو اليسار ووضع الناتج في المسجل$s2
	إذا كان $t1 > $s4 يتم وضع قيمة 1 في المسجل$t0 ، وإلا يتم وضع قيمة 0
	استخدمت تعليمة or هنا مع المسجل $0 بغرض نقل قيمة المسجل $s4 إلى المسجل$t1 ، ويمكن أن نستخدم بدلاً منها تعليمة add \$t1, \$0, \$s4

المطلوب:

1. ما هو تمثيل الذاكرة قبل تنفيذ تعليمات البرنامج؟
2. أُنشئ جدولاً لتتبع قيم المسجلات داخل الحلقة أثناء تنفيذ الكود.
3. ما هي قيمة المسجل $s5 بعد نهاية تنفيذ البرنامج؟
4. برأيك ما هي وظيفة البرنامج؟
5. ما هي التعديلات الواجب إجراءها على البرنامج في حال التعامل مع مصفوفة أنصاف كلمات؟
6. ما هي التعديلات الواجب إجراءها على البرنامج في حال التعامل مع مصفوفة بايتات؟

الحل:

1. **تمثيل الذاكرة قبل تنفيذ تعليمات البرنامج:**

ملاحظة :يجب تحويل القيم العشرية إلى ست عشرية قبل وضعها في تمثيل الذاكرة.

	+0	+4	+8	+c
0x1001000	0x1f	0x8	0xc	0xfffffffc
0x1001010	0x43	0x5	0xf	0x0

2. **قيم المسجلات:**

القيم الابتدائية قبل دخول الحلقة:

$s0 = 0x10010000 $t1 = 0x1F

تتبع قيم المسجلات أثناء المرور في الحلقة حتى الوصول إلى شرط التوقف والقفز إلى النهاية exit:

Rnd	1	2	3	4	5	6	7	8
$s2	0x0	0x4	0x8	0xc	0x10	0x14	0x18	0x1c
$s3	0x1001000	0x10010004	0x10010008	0x1001000c	0x10010010	0x10010014	0x10010018	0x1001001c
$s4	0x1f	0x8	0xc	0xfffffffc	0x43	0x5	0xf	0x0 (**exit**)
$t0	0x0	0x1	0x0	0x1	0x0	0x0	0x0	-------
$t1	0x1f	0x8	0x8	0xfffffffc	0xfffffffc	0xfffffffc	0xfffffffc	-------
$s6	0x1	0x2	0x3	0x4	0x5	0x6	0x7	-------

ملاحظة: سجلنا في الجدول قيمة كل مسجل في نهاية كل دورة من دورات الحلقة.

3. **قيمة المسجل $s5 بعد نهاية تنفيذ البرنامج هي 0xFFFFFFFC**، وتمثل أصغر قيمة في المصفوفة (الذاكرة).

4. **وظيفة البرنامج:**

وظيفة البرنامج هي المرور على عناصر مصفوفة (مواقع الذاكرة)، وإيجاد العنصر الأصغر فيها، والتوقف عن البحث عند مصادفة عنصر ذو قيمة 0.

5. التعديلات الواجب إجراؤها على البرنامج في حال التعامل مع مصفوفة أنصاف كلمات:

بعد التعديل	قبل التعديل
.half	.word
lh $t1, 0($s0)	lw $t1, 0($s0)
sll $s2, $s6, **1**	sll $s2, $s6, **2**
lh $s4, 0($s3)	lw $s4, 0($s3)

6. التعديلات الواجب إجراؤها على البرنامج في حال التعامل مع مصفوفة بايتات:

بعد التعديل	قبل التعديل
.byte	.word
lb $t1, 0($s0)	lw $t1, 0($s0)
تُحذف هذه التعليمة	sll $s2, $s6, **2**
add $s3, $s0, $s6	add $s3, $s0, $s2
lb $s4, 0($s3)	lw $s4, 0($s3)

ملاحظة هامة:

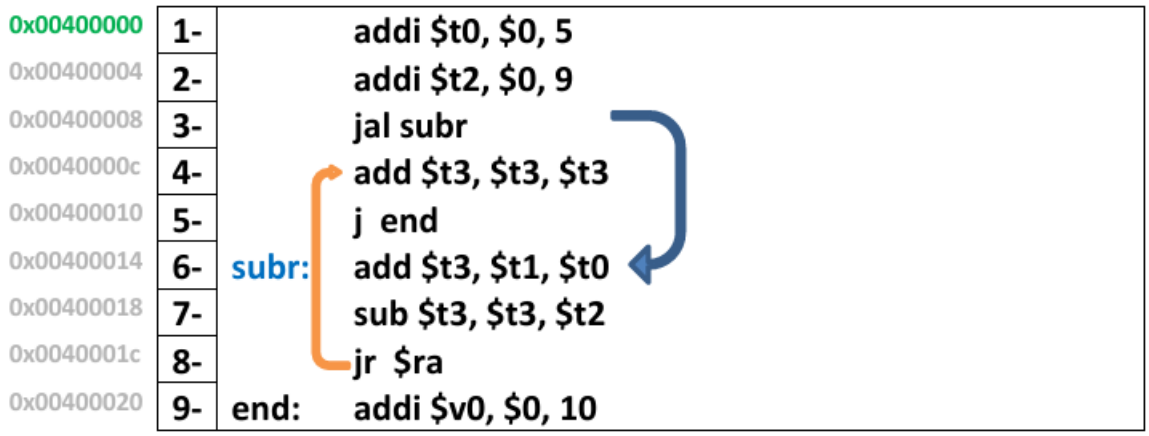
في الحالة العامة، إذا كان المطلوب هو تحويل البرنامج ليتعامل مع أنصاف كلمات أو بايتات، فتستخدم lh أو lb بدلاً من lw، كما فعلنا في مثالنا السابق.

أما في حالة خاصة، إذا كان المطلوب هو تحويل البرنامج ليتعامل مع أنصاف كلمات أو بايتات مع اعتبار أن الأعداد المخزنة في الذاكرة هي غير مؤشرة (unsigned) (أي اعتبارها جميعها موجبة)، فإننا نستخدم lhu أو lbu بدلاً من lw.

المثال 2:

"مثال بسيط عن القفز إلى البرنامج الفرعي والعودة منه"

لديك البرنامج المكتوب بلغة التجميع التالي:



شرح تعليمتي القفز:

- jal subr **(Jump And Link):**

تقوم هذه التعليمة بحفظ عنوان التعليمة التالية (التعليمة رقم 4) في المسجل الخاص $ra (return address)، أي يصبح $ra = 0x0040000C، ثم تنفذ القفز إلى العنوان subr لتنفيذ التعليمات الموجودة تحته.

- jr \$ra **(Jump Register):**

تقوم هذه التعليمة بالقفز إلى التعليمة ذات عنوان الذاكرة الموجود في المسجل $ra، أي العنوان x0040000C0، لتنفيذ التعليمات الموجودة بعده.

- يُسمى المسجل $ra **بمسجل عنوان العودة** (Return Address Register).
- وتسمى مجموعة التعليمات الموجودة تحت العنوان subr **بالبرنامج الفرعي** (Subroutine)، وهو مشابه للدالة (Function) في اللغات عالية المستوى.

المطلوب:

ما هي قيم المسجلات بعد تنفيذ البرنامج؟

الحل:

المسجل	القيمة
$t0	**0x5**
$t2	**0x9**
$t3	**0xe → 0xe → 0x1c**
$v0	**0xa**