المحاضرة 6 عملي معالجات صغرية – ميكاترونيكس

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

```
    .data
    .half 3, 5, 7, 7, 7, 7, 7, 7,7
    .text
    lui $s6, 0x1001
    addi $a1, $0, 7
    lui $v0, 0
    addi $s0, $0, 2
ss: sll $s1, $s0, 1
    add $t0, $s1, $s6
    lh $t1,0($t0)
    bne $t1, $a1, ex
    add $v0, $v0, $t1
    addi $s0, $s0, 1
    j ss
ex:
    sh $v0, 0($s6)
```

المطلوب:

1. ما هو تمثيل الذاكرة قبل تنفيذ تعليمات البرنامج؟
2. أُنشئ جدولاً لتتبع قيم المسجلات داخل الحلقة أثناء تنفيذ الكود.
3. ما هو تمثيل الذاكرة بعد نهاية تنفيذ البرنامج؟
4. كيف يمكن التعبير عن الكود السابق بلغة عالية المستوى؟
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.

	+2	+0	+6	+4	+a	+8	+e	+C
0x1001 0000	5	3	7	7	7	7	7	7
0x1001 0010	0	7	0	0	0	0	0	0
0x1001 0020	0	0	0	0	0	0	0	0

2. **قيم المسجلات:**

القيم الابتدائية قبل دخول الحلقة:

$s6 = 0x10010000 $s0 = 2 $a1 = 7 $v0 = 0

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

رقم الحلقة	1	2	3	4	5	6	7	8
$s1=$s0*2	4	6	8	10	12	14	16	18
$t0	0x10010004	0x10010006	0x10010008	0x1001000a	0x1001000c	0x1001000e	0x10010010	0x10010012
$t1	7	7	7	7	7	7	7	0
$t1 !=7 ?	×	×	×	×	×	×	×	break
$v0=$v0+$t1	7	14	21	28	35	42	49	-
$s0++	3	4	5	6	7	8	9	-

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

تكون قيم المسجلات النهائية:

$ v0 = 49 = 0x31 $s0 = 9 $t0 = 0x10010012

$t1 = 0 $s1 = 18 = 0x12

3. تمثيل الذاكرة بعد تنفيذ تعليمات البرنامج:

	+2	+0	+6	+4	+a	+8	+e	+C
0x1001 0000	5	49	7	7	7	7	7	7
0x1001 0010	0	7	0	0	0	0	0	0
0x1001 0020	0	0	0	0	0	0	0	0

4. الكود المماثل بلغة عالية المستوى:

```
int a[ ] = { 3,5,7,7,7,7,7,7,7,0,0, ….}
int sum=0
 for( int i=2 ;  ; i++)
     { if(a[i] !=7) break;
          else sum=sum+a[i] ; }
a[0]=sum;
```

5. 6. التعديلات الواجب إجراؤها على البرنامج للتعامل مع مصفوفة أنصاف كلمات /مصفوفة بايتات:

```
    .data
    .word 3, 5, 7, 7, 7, 7, 7, 7,7
    .text
    lui $s6, 0x1001
    addi $a1, $0, 7
    lui $v0, 0
    addi $s0, $0, 2
ss: sll $s1, $s0, 2
    add $t0, $s1, $s6
    lw $t1,0($t0)
    bne $t1, $a1, ex
    add $v0, $v0, $t1
    addi $s0, $s0, 1
    j ss
ex:
    sw $v0, 0($s6)
```

```
    .data
    .byte 3, 5, 7, 7, 7, 7, 7, 7,7
    .text
    lui $s6, 0x1001
    addi $a1, $0, 7
    lui $v0, 0
    addi $s0, $0, 2
ss:
    add $t0, $s0, $s6
    lb $t1,0($t0)
    bne $t1, $a1, ex
    add $v0, $v0, $t1
    addi $s0, $s0, 1
    j ss
ex:
    sb $v0, 0($s6)
```

المثال 2:

لديك البرنامج المكتوب بلغة التجميع التالي:

```
        .data
        .word 3, 5, 12, 6, 5, 8, 3, 9, 2, 7, 9, 8, 3
        .text
        lui $s0, 0x1001
        addi $t0, $0, 9
loop:   lw $t1, 0($s0)
        beq $t1, $t0, exit
        addi $v0, $v0, 1
        addi $s0, $s0, 4
        j loop
exit :
        sw $v0, 0($s0)
```

المطلوب:

1. ما هو تمثيل الذاكرة قبل تنفيذ تعليمات البرنامج؟
2. أُنشئ جدولاً لتتبع قيم المسجلات داخل الحلقة أثناء تنفيذ الكود.
3. ما هو تمثيل الذاكرة بعد نهاية تنفيذ البرنامج؟
4. ما هي وظيفة البرنامج؟
5. كيف يمكن التعبير عن الكود السابق بلغة عالية المستوى؟
6. ما هي قيمة المسجل $v0 لو أن مقدار التغير في المسجل $s0 ضمن الحلقة كان 8 بدلاً من 4؟
7. ما هي قيمة المسجل $v0 وتمثيل الذاكرة قبل وبعد التنفيذ لو أن القيم كانت كالآتي:

.word 2, 5, 1, 8, 5, 3, 1, 4, 6, 9, 0, 2

8. ما هي قيمة المسجل ذاته وتمثيل الذاكرة قبل وبعد عند القيم (.word 9, 5, 7, 1, 4, 9, 4)؟

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
0x1001 0000	3	5	12	6
0x1001 0010	5	8	3	9
0x1001 0020	2	7	9	8
0x1001 0030	3	0	0	0

2. **قيم المسجلات:**

القيم الابتدائية قبل دخول الحلقة: **$s0 = 0x10010000** **$t0 = 9**

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

رقم الحلقة	1	2	3	4	5	6	7	8
$t1	3	5	12	6	5	8	3	9
$t1=9 ?	×	×	×	×	×	×	×	break
$v0++	1	2	3	4	5	6	7	-
$s0+=4	0x10010004	0x10010008	0x1001000C	0x10010010	0x10010014	0x10010018	0x1001001C	-

3. **تمثيل الذاكرة بعد تنفيذ تعليمات البرنامج:**

	+0	+4	+8	+c
0x1001 0000	3	5	12	6
0x1001 0010	5	8	3	7
0x1001 0020	2	7	9	8
0x1001 0030	3	0	0	0

4. **وظيفة البرنامج** حساب عدد القيم التي تسبق ورود أول رقم 9 في مصفوفة ما، في هذا المثال هناك 7 قيم قبل ورود أول 9، نضع عددهم مكان أول 9.

5. **الكود المماثل بلغة عالية المستوى:**

```
int a[ ] = { 3, 5, 12, 6, 5, 8, 3, 9, 2, 7, 9, 8, 3, 0, 0 ….}
int n, c=0;
 for( int i=0 ;  ; i++)
        { n=i;
          if(a[i] == 9) break;
          else c++; }
a[n]=c;
```

6. **سيكون الحل $v0=5، وتمثيل الذاكرة بعد تنفيذ الكود:**

	+0	+4	+8	+c
0x1001 0000	3	5	12	6
0x1001 0010	5	8	3	9
0x1001 0020	2	7	5	8
0x1001 0030	3	0	0	0

7. **سيكون الحل $v0=9، وتمثيل الذاكرة قبل وبعد تنفيذ الكود سيكون نفسه:**

	+0	+4	+8	+c
0x1001 0000	2	5	1	8
0x1001 0010	5	3	1	4
0x1001 0020	6	9	0	2
0x1001 0030	0	0	0	0

8. **سيكون الحل $v0=0، وتمثيل الذاكرة بعد تنفيذ الكود:**

	+0	+4	+8	+c
0x1001 0000	0	5	7	1
0x1001 0010	4	9	4	0
	0	0	0	0
	0	0	0	0