Nguyễn Trọng Tuệ - 20194710 -VN03 – K64

Lab03's Assignments

1. Assignment 1

```
#Assignment 1 - Lab 03
.data
      i:    .word 2
      j:    .word 5
.text
      la    $t6, i                      #load address of i to register $t6
      lw    $s1, 0($t6)                 #load value of i ($t6) to $s1
      la    $t7, j                      #load address of j to register $t7
      lw    $s2, 0($t7)                 #load value of j ($t7) to $s2
   start:
      slt   $t0, $s2, $s1               # j < i => t0 = 1
      bne   $t0, $0, else               # if i <= j continue executing else
jumping to else
      addi  $t1, $t1, 1                 # x += 1
      addi  $t3, $0, 1                  # z = 1
      j     endif                       # jump to branch endif
   else:
      addi  $t2, $t2, -1                # y -= 1
      add   $t3, $t3, $t3               # z *= 2
   endif:
      syscall
```

- Các giá trị được gán vào các thanh ghi:
  - i: thanh ghi $s1, i = 2
  - j: thanh ghi $s2, j = 5
  - x: thanh ghi $t1, mặc định = 0
  - y: thang ghi $t2, mặc định = 0
  - z: thanh ghi $t3, mặc định = 0

- $t0: thanh ghi chứa kết quả của phép so sánh (i <= j)

- Kết quả sau khi thực hiện chương trình:

$t0	8	0x00000000
$t1	9	0x00000001
$t2	10	0x00000000
$t3	11	0x00000001

- $s1 < $s2 do 2 < 5 => $t0 = 0 *(slt $t0, $s2, $s1)*. Do đó, *bne $t0, $0, else* không thực hiện, các câu lệnh tiếp theo đến *j endif* được thực hiện
- Kết quả: x($t1) = 0 + 1 = 1
  y($t2) = 0 (không thực hiện)
  z($t3) = 1 (= $0 + 1)

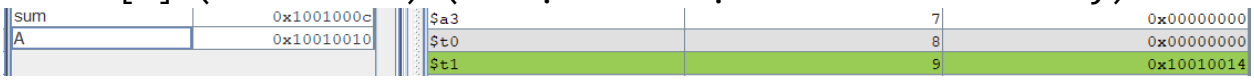
2. *Assignment 2*

```
#Assignment 2 - Lab 03
.data
   i:          .word 0
   n:          .word 6
   step:       .word 1
   sum:        .word 0
   A:          .word 1, 2, 3, 4, 5, 6
.text
   la    $t7, i
   lw    $s1, 0($t7)
   la    $s2,  A
   la    $t7,  n
   lw    $s3, 0($t7)
   la    $t7,  step
   lw    $s4, 0($t7)
   la    $t7,  sum
   lw    $s5, 0($t7)
loop:
   add   $s1, $s1, $s4
   add   $t1, $s1, $s1
   add   $t1, $t1, $t1
   add   $t1, $t1, $s2

   lw    $t0, 0($t1)
   add   $s5, $s5, $t0
   bne   $s1, $s3, loop

   syscall
```

- Các thanh ghi được gán giá trị:
  - $s1: i
  - $s2: địa chỉ của mảng A (địa chỉ phần tử đầu tiên của A). Mảng A gồm 6 phần tử {1, 2, …, 6}
  - $s3: n (hằng số so sánh n)
  - $s4: step (bước nhảy)
  - $s5: sum (tổng)
- Kết quả sau khi thực hiện chương trình:

$t0	8	0x00000000
$t1	9	0x10010028
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x1001000c
$s0	16	0x00000000
$s1	17	0x00000006
$s2	18	0x10010010
$s3	19	0x00000006
$s4	20	0x00000001
$s5	21	0x00000014
$s6	22	0x00000000
$s7	23	0x00000000

  - Kết quả là 14_{hex} = 20 thay vì 15_{hex} = 21. Sở dĩ xảy ra điều này là do chương trình thực hiện việc cộng chỉ số i với step (1) ngay từ đầu nên phần tử tổng là sum sẽ được tính từ phần tử A[i + 1] (0x10010014) thay vì từ A[i] (0x10010010) (i được khởi tạo = 0 ở bài toán này).



  - Bài toán kết thúc tại $t0 = 0x10010028 = A[6] = 0 (không được khởi tạo) mà mảng chỉ có phần tử cuối cùng là A[5]. Trong trường hợp đằng sau mảng A ở đây có phần tử chứa giá trị khác sẽ dễ gây ra nhầm lẫn về mặt tính toán

3. *Assignment 3*

```
#Assignment 3 - Lab 03
.data
	test:		.word		1
```

```
.text
      la    $s0, test          #load the address of test variable
      lw    $s1, 0($s0)        #load the value of test to register $s1
      li    $t0, 0             #load value for test case
      li    $t1, 1
      li    $t2, 2
      beq   $s1, $t0, case_0
      beq   $s1, $t1, case_1
      beq   $s1, $t2, case_2
      j     default
   case_0:
      addi  $s2, $s2, 1              # a = a+1
      j     continue
   case_1:
      sub $s2, $s2, $t1              # a = a-1
      j     continue
   case_2:
      add $s3, $s3, $s3              # b = 2*b
      j     continue
   default:
   continue:
      syscall
```

- Các thanh ghi được gán giá trị:
  - $s0: địa chỉ của biến test
  - $s1: giá trị của biến test
  - $t0: case 0
  - $t1: case 1
  - $t2: case 2
  - $s2: biến a, mặc định = 0
  - $s3: biến b, mặc định = 0
- Kết quả của từng test case với biến kiểu nguyên test thay đổi giá trị trong khoảng [0; 2]
  - test = 0 => a += 1 = 0 + 1 = 1 ⇔ $s2 = 1 = 0x00000001
    $s3 = 0 = 0x00000000

$s0	16	0x10010000
$s1	17	0x00000000
$s2	18	0x00000001
$s3	19	0x00000000

  - test = 1 =>a -= 1 = 0 - 1 = -1 ⇔ $s2 = -1 = 0xffffffff
    $s3 = 0 = 0x00000000

$s0	16	0x10010000
$s1	17	0x00000001
$s2	18	0xffffffff
$s3	19	0x00000000

- test = 2 => b *= 2 = 0 * 2 = 0 ⇔ $s2 = 0 = 0x00000000
  $s3 = 0 = 0x00000000

$s0	16	0x10010000
$s1	17	0x00000002
$s2	18	0x00000000
$s3	19	0x00000000

4. *Assignment 4*

a. i < j

```
#Assignment 4a - Lab 03

   .data
      i:    .word 2
      j:    .word 5
      x:    .word 10
      y:    .word 12
      z:    .word 6
   .text
      la    $t6, i                        #load address of i to
   register $t6
      lw    $s1, 0($t6)       #load value of i ($t6) to $s1
      la    $t7, j                        #load address of j to
   register $t7
      lw    $s2, 0($t7)       #load value of j ($t7) to $s2
      la    $t5, x                  #load address of x to register $t5
      lw    $t1, 0($t5)       #load value of x ($t5) to $t1
      la    $t5, y                  #load address of y to register $t5
      lw    $t2, 0($t5)       #load value of y($t5) to $t2
      la    $t5, z                  #load address of z to register $t5
      lw    $t3, 0($t5)       #load value of z ($t5) to $t3
      start:
      slt   $t0, $s1, $s2           # i < j => t0 = 1
      beq   $t0, $0, else           # if i < j (t0 = 1)  continue
   executing else i >= j (t0 = 0)  jumping to else
      addi  $t1, $t1, 1       # x += 1
      addi $t3, $0, 1               # z = 1
```

```
            j     endif             # jump to branch endif
      else:
      addi  $t2, $t2, -1            # y -= 1
      add   $t3, $t3, $t3           # z *= 2
      endif:
      syscall
```

b. i >= j

```
#Assignment 4b - Lab 03

   .data
      i:    .word 7
      j:    .word 5
      x:    .word 10
      y:    .word 12
      z:    .word 6
   .text
      la    $t6, i                        #load address of i to
   register $t6
      lw    $s1, 0($t6)       #load value of i ($t6) to $s1
      la    $t7, j                        #load address of j to
   register $t7
      lw    $s2, 0($t7)       #load value of j ($t7) to $s2
      la    $t5, x                #load address of x to register $t5
      lw    $t1, 0($t5)       #load value of x ($t5) to $t1
      la    $t5, y                #load address of y to register $t5
      lw    $t2, 0($t5)       #load value of y($t5) to $t2
      la    $t5, z                #load address of z to register $t5
      lw    $t3, 0($t5)       #load value of z ($t5) to $t3
      start:
      slt   $t0, $s1, $s2           # i < j => t0 = 1 <=> ( i >= j =>
   t0 = 0)
      bne   $t0, $0, else           # if i >= j (t0 == 0)  continue
   executing else i < j (t0 == 1 != 0)  jumping to else
      addi  $t1, $t1, 1       # x += 1
      addi $t3, $0, 1               # z = 1
            j     endif             # jump to branch endif
      else:
      addi  $t2, $t2, -1            # y -= 1
      add   $t3, $t3, $t3           # z *= 2
      endif:
      syscall
```

c. i+j <= 0

```
#Assignment 4c - Lab 03
```

```
   .data
      i:    .word 2
      j:    .word 5
      x:    .word 10
      y:    .word 12
      z:    .word 6
   .text
      la    $t6, i                        #load address of i to
   register $t6
      lw    $s1, 0($t6)       #load value of i ($t6) to $s1
      la    $t7, j                        #load address of hj to
   register $t7
      lw    $s2, 0($t7)       #load value of j ($t7) to $s2
      la    $t5, x                #load address of x to register $t5
      lw    $t1, 0($t5)       #load value of x ($t5) to $t1
      la    $t5, y                #load address of y to register $t5
      lw    $t2, 0($t5)       #load value of y($t5) to $t2
      la    $t5, z                #load address of z to register $t5
      lw    $t3, 0($t5)       #load value of z ($t5) to $t3
      start:
      add   $t4, $s1, $s2           # t4 = i + j
      slt   $t0, $0, $t4            # 0 < (i + j) then t0 = 1 else (i +
   j) <= 0 then t0 = 0
      bne   $t0, $0, else           # if (i + j) <= 0 (t0 = 0)
   continue executing else (i + j) > 0 (t0 = 1 != 0)  jumping to else
      addi  $t1, $t1, 1       # x += 1
      addi $t3, $0, 1               # z = 1
            j     endif             # jump to branch endif
      else:
      addi  $t2, $t2, -1            # y -= 1
      add   $t3, $t3, $t3           # z *= 2
      endif:
      syscall
```

d. i+j > m+n

```
#Assignment 4d - Lab 03

   .data
      i:    .word 2
      j:    .word 5
      m:    .word 4
      n:    .word 1
      x:    .word 10
      y:    .word 12
      z:    .word 6
   .text
```

```
   la    $t6, i                        #load address of i to
register $t6
   lw    $s1, 0($t6)       #load value of i ($t6) to $s1
   la    $t7, j                        #load address of i to
register $t7
   lw    $s2, 0($t7)       #load value of j ($t7) to $s2
   la    $t6, m                 #load address of m to register $t6
   lw    $s3, 0($t6)       #load value of m ($t6) to $s1
   la    $t7, n                 #load address of n to register $t7
   lw    $s4, 0($t7)       #load value of n ($t7) to $s2
   la    $t6, i                        #load address of i to
register $t6
   lw    $s1, 0($t6)       #load value of i ($t6) to $s1
   la    $t7 j                  #load address of i to register $t7
   lw    $s2, 0($t7)       #load value of j ($t7) to $s2
   la    $t5, x                 #load address of x to register $t5
   lw    $t1, 0($t5)       #load value of x ($t5) to $t1
   la    $t5, y                 #load address of y to register $t5
   lw    $t2, 0($t5)       #load value of y($t5) to $t2
   la    $t5, z                 #load address of z to register $t5
   lw    $t3, 0($t5)       #load value of z ($t5) to $t3
   start:
   add   $t4, $s1, $s2           # t4 = i + j
   add   $t5, $s3, $s4           # t5 = m + n
   slt   $t0, $t5, $t4           # (m + n) < (i + j) then t0 = 1
else (i + j) <= (m + n) then t0 = 0
   beq   $t0, $0, else           # if (m + n) < (i + j) (t0 = 1 !=
0) then continue executing else (i + j) >=  (m + n) (t0 = 0)
jumping to else
   addi  $t1, $t1, 1       # x += 1
   addi $t3, $0, 1               # z = 1
         j     endif             # jump to branch endif
   else:
   addi  $t2, $t2, -1            # y -= 1
   add   $t3, $t3, $t3           # z *= 2
   endif:
   syscall
```

5. *Assignment 5*

a. i < n

```
#Assignment 5a - Lab 03

   .data
   i:          .word 0
```

```
      n:          .word 4
      step:       .word 1
      sum:        .word 0
      A:          .word 1, 2, 3, 4, 5, 6
   .text
      la    $t7, i
      lw    $s1, 0($t7)
      la    $s2,  A
      la    $t7,  n
      lw    $s3, 0($t7)
      la    $t7,  step
      lw    $s4, 0($t7)
      la    $t7,  sum
      lw    $s5, 0($t7)

      loop:
            add   $s1, $s1, $s4
            add   $t1, $s1, $s1
            add   $t1, $t1, $t1
            add   $t1, $t1, $s2

            lw    $t0, 0($t1)
            add   $s5, $s5, $t0
            slt   $t2, $s1, $s3
            bne   $t2, $0, loop
            syscall
```

b. i <= n

```
#Assignment 5b - Lab 03

.data

      i:          .word 0

      n:          .word 4

      step:       .word 1

      sum:        .word 0

      A:          .word 1, 2, 3, 4, 5, 6

.text

      la    $t7, i

      lw    $s1, 0($t7)
```

```
      la    $s2,  A
      la    $t7,  n
      lw    $s3, 0($t7)
      la    $t7,  step
      lw    $s4, 0($t7)
      la    $t7,  sum
      lw    $s5, 0($t7)

   loop:
      add   $s1, $s1, $s4
      add   $t1, $s1, $s1
      add   $t1, $t1, $t1
      add   $t1, $t1, $s2

      lw    $t0, 0($t1)
      add   $s5, $s5, $t0
      slt   $t2, $s3, $s1
      beq   $t2, $0, loop
      syscall
```

c. sum >= 0

```
#Assignment 5c - Lab 03
   .data
      i:          .word 0
      n:          .word 4
      step:       .word 1
      sum:        .word 0
      A:          .word 0, -8, 2, 7, 5, 6
   .text
```

```
      la    $t7, i
      lw    $s1, 0($t7)
      la    $s2,  A
      la    $t7,  n
      lw    $s3, 0($t7)
      la    $t7,  step
      lw    $s4, 0($t7)
      la    $t7,  sum
      lw    $s5, 0($t7)

      loop:
            add   $s1, $s1, $s4
            add   $t1, $s1, $s1
            add   $t1, $t1, $t1
            add   $t1, $t1, $s2

            lw    $t0, 0($t1)
            add   $s5, $s5, $t0
            slt   $t2, $s5, $0
            bne   $t2, $0, loop
            syscall
```

d. A[i] = 0

```
#Assignment 5d - Lab 03

   .data
      i:          .word 0
      n:          .word 4
      step:       .word 1
      sum:        .word 0
      A:          .word 1, 2, 3, 0, 5, 6
   .text
      la    $t7, i
      lw    $s1, 0($t7)
      la    $s2,  A
      la    $t7,  n
      lw    $s3, 0($t7)
      la    $t7,  step
      lw    $s4, 0($t7)
      la    $t7,  sum
      lw    $s5, 0($t7)

      loop:
            add   $s1, $s1, $s4
            add   $t1, $s1, $s1
            add   $t1, $t1, $t1
```

```
                add   $t1, $t1, $s2

                lw    $t0, 0($t1)
                add   $s5, $s5, $t0
                bne   $t0, $0, loop
                syscall
```

6. Assignment 6

```
# Assignment 6 - Lab 03

   .data
      i:          .word 0
      n:          .word 6
      max:        .word 0
      A:          .word 16, 2, 3, 4, 5, 32
   .text
      la    $t7, i
      lw    $s1, 0($t7)
      la    $s2,  A
      la    $t7,  n
      lw    $s3, 0($t7)
      la    $t7,  max
      lw    $s5, 0($t7)

      loop:
            beq   $s1, $s3, endloop
            sll   $t1, $s1, 2
            add   $t1, $t1, $s2

            lw    $t0, 0($t1)
            slt   $t2, $s5, $t0
            beq   $t2, $0, next
            add   $s5, $0, $t0
      next:
            addi $s1, $s1, 1
            j     loop
      endloop:
            syscall
```