BÁO CÁO TUẦN 6

Lê Việt Hùng – 20194580

Bài 1:

hom1.asm | home2.asm

```
1  .data
2  A: .word -2, 6, -1, 3, -2
3  .text
4  main: la $a0,A
5  li $a1,5
6  j mspfx
7  nop
8  continue:
9  lock: j lock
10 nop
11 end_of_main:
12 #-----------------------------------------------------------------
13 #Procedure mspfx
14 # @brief find the maximum-sum prefix in a list of integers
15 # @param[in] a0 the base address of this list(A) need to be processed
16 # @param[in] a1 the number of elements in list(A)
17 # @param[out] v0 the length of sub-array of A in which max sum reachs.
18 # @param[out] v1 the max sum of a certain sub-array
19 #-----------------------------------------------------------------
20 #Procedure mspfx
21 #function: find the maximum-sum prefix in a list of integers
22 #the base address of this list(A) in $a0 and the number of
23 #elements is stored in a1
24 mspfx: addi $v0,$zero,0 #initialize length in $v0 to 0
25 addi $v1,$zero,0 #initialize max sum in $v1to 0
26 addi $t0,$zero,0 #initialize index i in $t0 to 0
27 addi $t1,$zero,0 #initialize running sum in $t1 to 0
28 loop: add $t2,$t0,$t0 #put 2i in $t2
```

```
24  mspfx: addi $v0,$zero,0 #initialize length in $v0 to 0
25  addi $v1,$zero,0 #initialize max sum in $v1to 0
26  addi $t0,$zero,0 #initialize index i in $t0 to 0
27  addi $t1,$zero,0 #initialize running sum in $t1 to 0
28  loop: add $t2,$t0,$t0 #put 2i in $t2
29  add $t2,$t2,$t2 #put 4i in $t2
30  add $t3,$t2,$a0 #put 4i+A (address of A[i]) in $t3
31  lw $t4,0($t3) #load A[i] from mem(t3) into $t4
32  add $t1,$t1,$t4 #add A[i] to running sum in $t1
33  slt $t5,$v1,$t1 #set $t5 to 1 if max sum < new sum
34  bne $t5,$zero,mdfy #if max sum is less, modify results
35  j test #done?
36  mdfy: addi $v0,$t0,1 #new max-sum prefix has length i+1
37  addi $v1,$t1,0 #new max sum is the running sum
38  test: addi $t0,$t0,1 #advance the index i
39  slt $t5,$t0,$a1 #set $t5 to 1 if i<n
40  bne $t5,$zero,loop #repeat if i<n
41  done: j continue
42  mspfx_end:
```

Kết quả:

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	4
$v1	3	6
$a0	4	268500992
$a1	5	5
$a2	6	0

Giải thích:

.data:

- khai báo mảng A[] = { -2, 6, -1, 3, -2}

.text:

- La: gán địa chỉ mảng A vào $a0
- Li: Gán $s1 = 5 = số phần tử của mảng
- Jump: thực hiện chương trình con MSPFX
- Lock: thực hiện vòng lặp vô hạn để chương trình không chạy lại lệnh ở dưới

*Chương trình con MSPFX (Tìm tổng prefix lớn nhất trong mảng integer):

- Addi: gán $v0 = 0 (độ dài của dãy prefix có tổng lớn nhất)
- addi: Gán $v1 = 0 (tổng của dãy prefix lớn nhất)
- addi: Gán $t0 = 0 (Gán chỉ mục i = 0)
- addi: Gán $t1 = 0 (Gán tổng hiện thời (running sum) = 0)
- Nhãn loop:
    - add: $t2 = 2i
    - add: $t2 = 4i
    - add: $t3 = A + 4i = địa chỉ của A[i]
    - lw: $t4 = giá trị của địa chỉ $t3 = giá trị của A[i]
    - add: $t1 = $t1 + $t4 (tính tổng giá trị hiện thời)
    - slt: kiểm tra max sum < running sum không? Nếu đúng thì gán = 1
    - bne: Nếu max sum < running sum thì nhảy đến mdfy (thay đổi giá trị max sum)
    - j : Nhảy đến test
- Nhãn mdfy:
    - addi: $v0 = $t0 + 1 (vì độ dài của dãy = chỉ số i + 1)
    - addi: $v1 = $t1 + 0 (Gán giá trị max mới vào $v1)
- Nhãn test:
    - addi: $t0 = $t0 + 1 (I = I + 1)
    - slt: So sánh $t0 < $a1 ?
    - bne: Nếu I < n thì lặp lại (nhảy vào nhãn loop)
    - Nếu không thì nhảy đến nhãn continue và vào nhãn lock: lặp vô tận

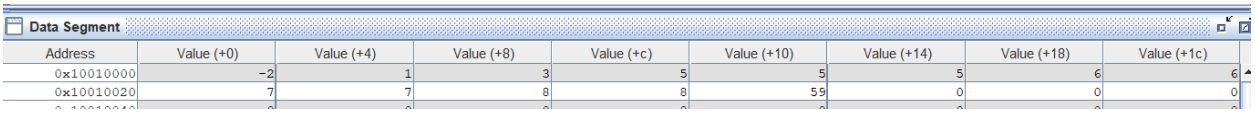
Bài 2:

hom1.asm | home2.asm

```
.data
A: .word 7, -2, 5, 1, 5,6,7,3,6,8,8,59,5
Aend: .word
.text
main: la $a0,A #$a0 = Address(A[0])
la $a1,Aend
addi $a1,$a1,-4 #$a1 = Address(A[n-1])
j sort #sort
after_sort: li $v0, 10 #exit
syscall
end_main:
#------------------------------------------------------------------
#procedure sort (ascending selection sort using pointer)
#register usage in sort program
#$a0 pointer to the first element in unsorted part
#$a1 pointer to the last element in unsorted part
#$t0 temporary place for value of last element
#$v0 pointer to max element in unsorted part
#$v1 value of max element in unsorted part
#------------------------------------------------------------------
sort: beq $a0,$a1,done #single element list is sorted
j max #call the max procedure
after_max: lw $t0,0($a1) #load last element into $t0
sw $t0,0($v0) #copy last element to max location
sw $v1,0($a1) #copy max value to last element
addi $a1,$a1,-4 #decrement pointer to last element
j sort #repeat sort for smaller list
done: j after sort
```

```
26  addi $a1,$a1,-4 #decrement pointer to last element
27  j sort #repeat sort for smaller list
28  done: j after_sort
29  #------------------------------------------------------------------
30  #Procedure max
31  #function: fax the value and address of max element in the list
32  #$a0 pointer to first element
33  #$a1 pointer to last element
34  #------------------------------------------------------------------
35  max:
36  addi $v0,$a0,0 #init max pointer to first element
37  lw $v1,0($v0) #init max value to first value
38  addi $t0,$a0,0 #init next pointer to first
39  loop:
40  beq $t0,$a1,ret #if next=last, return
41  addi $t0,$t0,4 #advance to next element
42  lw $t1,0($t0) #load next element into $t1
43  slt $t2,$t1,$v1 #(next)<(max) ?
44  bne $t2,$zero,loop #if (next)<(max), repeat
45  addi $v0,$t0,0 #next element is new max element
46  addi $v1,$t1,0 #next value is new max value
47  j loop #change completed; now repeat
48  ret:
49  j after_max
```

Kết quả:



Giải thích:

.data:

- Khai báo mảng A = { 7, -2, 5, 1, 5,6,7,3,6,8,8,59,5}
- Khai báo số cuối của mảng A unsorted : Aend

.text:

- La: gán $a0 = địa chỉ của mảng A = A[0]

- Addi: $a1 = $a1 – 4 (Gán $a1 là địa chỉ của A[n- 1] = Địa chỉ của phần tử cuối cùng của mảng. Vì $a1 được khai bao ngay sau khi khai báo mảng A nên địa chỉ của $a1 nằm ngay sau phần tử cuối của mảng. Vì vậy, giảm đi 4 bit là địa chỉ của phần tử cuối của mảng.
- J: Nhảy đến nhãn SORT
- After_sort : Thực hiện thoát chương trình bằng lệnh syscall

- Nhãn sort:
- Beq: So sánh địa chỉ của $a0, $s1, nếu bằng nhau tức là đã sắp xếp xong (phần tử cuối của mảng A unsorted trùng với phần tử đầu)
- Nếu không nhảy đến nhãn MAX: tìm phần tử lớn nhất
- AFTER_MAX: sau khi tìm được phần tử lớn nhất, thực hiện đổi chỗ phần tử Aend với phần tử max và lùi phần tử Aend
- Lw: Lưu địa chỉ của Aend vào $t0
- Sw: Gán địa chỉ phần tử Aend vào địa chỉ của phần tử lớn nhất
- Sw: Gán giá trị của phần tử lớn nhất vào phần tử Aend
- Addi: giảm Aend đi 1 phần tử
- J: Nhảy về nhãn sort. Thực hiện vòng lặp
- Nhãn MAX:
- Addi: Gán $v0 = A[0] (Gán con trỏ trỏ đến phần tử lớn nhất vào A[0])
- Lw: Gán $v1 = A[0] (Gán $v1 là giá trị lớn nhất bằng A[0])
- Addi: Khai báo con trỏ ‘next’ bằng A[0]
- Nhãn LOOP:
    - – beq : Nếu địa chỉ con trỏ kế tiếp = địa chỉ con trỏ cuối cùng thì return
    - – addi: Tăng địa chỉ của con trỏ ‘next’ lên 4 bit (Trỏ vào phần tử kế tiếp)
    - Lw: lưu giá trị của phần tử kế tiếp vào $t1
    - Slt: So sánh giá trị phần tử kế tiếp $t1 < giá trị của max?
    - Bne: Nếu không thì quay lại loop
    - Addi: Nếu có thì gán địa chỉ phần tử kế tiếp là phần tử max mới
    - Addi: Gán giá trị max = giá trị của phần tử mới.

- Yêu cầu 2: Đảo chiều sắp xếp

```
slt $t2,$v1,$t1 # (min) < (next) ?
bne $t2,0,loop #if (min)<(next, repeat
```

Thay đổi vị trí $v1, $t1 tại lệnh slt

Hoặc giữ nguyên thay 0 thành 1 ở lệnh bne.

Bài 3:

- Chiều tăng dần

home2.asm as3.asm

```
  .data
arr: .word 10, 60, 40, 70, 20, 30, 90, 100, 0, 80, 50
  space: .asciiz " "
  .text
  .globl main

main:
  lui $s0, 0x1001                  #arr[0]
  li $t0, 0                        #i = 0
  li $t1, 0                        #j = 0
  li $s1, 11                       #n = 11
  li $s2, 11                       #n-i cho vòng lặp inner_loop
  add $t2, $zero, $s0              #cho địa chỉ vòng lặp bởi i
  add $t3, $zero, $s0              #cho địa chỉ vòng lặp bởi j

  addi $s1, $s1, -1

outer_loop:
  li  $t1, 0                       #j = 0
  addi $s2, $s2, -1                #Giảm size cho vòng lặp inner_loop
  add $t3, $zero, $s0              #khởi tạo lại địa chỉ vòng lặp j

  inner_loop:
    lw $s3, 0($t3)                 #arr[j]
    addi $t3, $t3, 4               #địa chỉ vòng lặp j += 4
```

```
  inner_loop:
    lw $s3, 0($t3)                   #arr[j]
    addi $t3, $t3, 4                 #địa chỉ vòng lặp j += 4
    lw $s4, 0($t3)                   #arr[j+1]
    addi $t1, $t1, 1                 #j++

    slt $t4, $s3, $s4                #đặt $t4 = 1 nếu $s3 < $s4
    bne $t4, $zero, cond
    swap:
      sw $s3, 0($t3)               #A[j+1] = s3
      sw $s4, -4($t3)              #A[j] = s4
      #lw $s4, 0($t3)              #s4 = A[j+1] = s3

    cond:
      bne $t1, $s2, inner_loop       #j != n-i

    addi $t0, $t0, 1                   #i++
  bne $t0, $s1, outer_loop           #i != n

  li $t0, 0
  addi $s1, $s1, 1
print_loop:
  li $v0, 1
  lw $a0, 0($t2)
```

```
43    addi $s1, $s1, 1
44  print_loop:
45    li $v0, 1
46    lw $a0, 0($t2)
47    syscall
48    li $v0, 4
49    la $a0, space
50    syscall
51
52    addi $t2, $t2, 4                  #địa chỉ vòng lặp i += 4
53    addi $t0, $t0, 1                  #i++
54    bne $t0, $s1, print_loop          #i != n
55
56  exit:
57    li $v0, 10
58    syscall
```

Line: 34 Column: 8 ☑ Show Line Numbers

Kết quả:

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	10	20	30	40	50	60	70
0x10010020	80	90	100	32	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0

- Chiều giảm dần

```
slt $t4, $s4      , $s3          #đặt $t4 = 1 nếu $s4 < $s3
bne $t4, $zero, cond
```

Kết quả:

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	100	90	80	70	60	50	40	30
0x10010020	20	10	0	32	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0

Bài 4:

```
.data

array:          .word           10, 60, 40, 70, 20, 30, 90, 100, 0, 80, 50
size:           .word           11

.text
.globl main
main:
sort_prep:
                la              $t0, array                      # $t0 = A[0]
                lw              $t1, size                       # $t1 = 11
                li              $t2, 1                          # i = 1
sort_xloop:
                la              $t0, array                      # $t0 = A[0]
                bge             $t2, $t1, sort_xloop_end        # while ($t2 < $t1).
                move            $t3, $t2                        # copy $t2 to $t3.
sort_iloop:
                la              $t0, array                      # $t0 = A[0]
                mul             $t5, $t3, 4                     # $t5 = $t3 * 4
                add             $t0, $t0, $t5                   # nhay den A[i]
                ble             $t3, $zero, sort_iloop_end      # while (t3 > 0).
                lw              $t7, 0($t0)                     # $t7 = A[i]
                lw              $t6, -4($t0)                    # $t6 = A[i-1]
                bge             $t7, $t6, sort_iloop_end        # while (A[i] < A[i-1]).
                lw              $t4, 0($t0)                     #swap
                sw              $t6, 0($t0)                     #swap
                sw              $t4, -4($t0)                    #swap
                subi            $t3, $t3, 1
                j               sort_iloop
sort_iloop_end:
                addi            $t2, $t2, 1                     # i += 1.
                j               sort_xloop
sort_xloop_end:
exit:
                li              $v0, 10
                syscall
```

Kết quả:

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	10	20	30	40	50	60	70
0x10010020	80	90	100	11	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0

- Chiều ngược lại:

```
bge             $t6,$t7, sort_iloop_end         # while (A[i  - 1] < A[i]).
lw              $t4, 0($t0)                     #swap
```

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	100	90	80	70	60	50	40	30
0x10010020	20	10	0	11	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0