

Họ và tên: Bùi Văn Anh

MSSV: 20184026

Học phần: Thực hành kiến trúc máy tính

Mã lớp 122032

Báo cáo Lab06

1. Assignment 1

Sample Code 1: tìm prefix-sum lớn nhất của một mảng số nguyên A: -5, -3, 5, -1, 6, 7, -8

```

1  .data
2  A: .word -5,-3,5,-1,6,7,-8
3  .text
4  main:    la $a0,A
5           li $a1,7
6           j mspfx
7           nop
8  continue:
9  lock:    j mspfx_end
10          nop
11  end_of_main:

```

Địa chỉ mảng A lưu vào thanh ghi \$a0, số lượng phần tử của mảng lưu vào \$a1

Kết quả: \$a0 = 0x10010000, \$a1 = 0x00000007

Sau lệnh j mspfx, pc = 0x0040001c = địa chỉ của nhãn mspfx -> chương trình chuyển đến chạy lệnh tại mspfx

mspfx tìm prefix-sum lớn nhất của mảng

```

24  mspfx:   addi $v0,$zero,0 #initialize the length in $v0 to 0
25          addi $v1,$zero,0 #initialize the max sum in $v1 to 0
26          addi $t0,$zero,0 #initialize the index i in $t0 to 0
27          addi $t1,$zero,0 #initialize the 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 the running sum in $t1
33          slt $t5,$v1,$t1 #set $t5 to 1 if max sum < new sum
34          bne $t5,$zero,mdfy #if the 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 #increment 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:

```

\$v0: độ dài của prefix/ \$v1: prefix-sum lớn nhất/ \$t0: index i/ \$t1: prefix-sum đang tính.

loop:

Do 1 word 4 byte -> địa chỉ A[i] và A[i+1] chênh nhau 4 byte -> \$t2 = 4*i

-> \$t3 lưu địa chỉ của A[i]

-> lưu word có địa chỉ lưu ở \$t3 vào \$t4

-> sum = sum + A[i] -> tổng của prefix đang tính lưu vào \$t1

-> so sánh prefix-sum đang tính được lưu ở \$t1 với max prefix-sum lưu ở \$v1

- Nếu \$t1 > \$v1 (slt trả về 1) thì chuyển đến nhãn mdfy
mdfy: tăng độ dài của prefix max lên 1, gán giá trị \$t1 vào \$v1
Sau đó chạy tiếp test
- Nếu \$t1 < \$v1 bỏ qua mdfy, chạy đến nhãn test

Test: Tăng i lên 1

-> So sánh i với n, nếu i < n (\$t0 < \$a1) chạy lại nhãn loop

Nếu i > n -> nhảy đến continue => kết thúc mspfx.

Ví dụ sau loop đầu tiên (i=0) giá của các thanh ghi

Registers	Coproc 1	Coproc 0	
Name	Number		Value
\$zero	0		0x00000000
\$at	1		0x10010000
\$v0	2		0x00000000
\$v1	3		0x00000000
\$a0	4		0x10010000
\$a1	5		0x00000007
\$a2	6		0x00000000
\$a3	7		0x00000000
\$t0	8		0x00000001
\$t1	9		0xffffffffb
\$t2	10		0x00000000
\$t3	11		0x10010000
\$t4	12		0xffffffffb
\$t5	13		0x00000001
\$t6	14		0x00000000
\$t7	15		0x00000000
\$s0	16		0x00000000
\$s1	17		0x00000000
\$s2	18		0x00000000
\$s3	19		0x00000000
\$s4	20		0x00000000
\$s5	21		0x00000000
\$s6	22		0x00000000
\$s7	23		0x00000000
\$t8	24		0x00000000
\$t9	25		0x00000000
\$k0	26		0x00000000
\$k1	27		0x00000000
\$gp	28		0x10008000
\$sp	29		0x7fffffc
\$fp	30		0x00000000
\$ra	31		0x00000000
pc			0x0040002c
hi			0x00000000
lo			0x00000000

i=\$t0=1 < n => quay lại nhãn loop

Final result:

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000006
\$v1	3	0x00000009
\$a0	4	0x10010000
\$a1	5	0x00000007
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000007
\$t1	9	0x00000001
\$t2	10	0x00000018
\$t3	11	0x10010018
\$t4	12	0xffffffff8
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400014
hi		0x00000000
lo		0x00000000

prefix-sum lớn nhất là giá trị \$v1 = 9 là prefix của \$v0 = 6 phần tử đầu.

2. Assignment 2

Sample code 2 sắp xếp mảng theo thứ tự tăng dần bằng thuật toán selection sort:

Đi tìm phần tử có giá trị lớn nhất trong đoạn đoạn chưa được sắp xếp và đổi cho phần tử lớn nhất đó với phần tử ở cuối đoạn chưa được sắp xếp. Thuật toán sẽ chia mảng làm 2 mảng con

1. Một mảng con đã được sắp xếp
2. Một mảng con chưa được sắp xếp

Tại mỗi bước lặp của thuật toán, phần tử lớn nhất ở mảng con chưa được sắp xếp sẽ được di chuyển về đoạn đã sắp xếp.

```

.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:

```

\$a0 lưu địa chỉ của mảng A/ \$a1 lưu địa chỉ phần tử cuối cùng mảng A

```

20 sort:    beq $a0,$a1,done #single-element list is sorted
21          j max #call max procedure
22 after_max: lw $t0,0($a1) #load last element into $t0
23            sw $t0,0($v0) #copy last element to max location
24            sw $v1,0($a1) #copy max value to last element
25            addi $a1,$a1,-4 #decrement pointer to the last
26 element:
27            j sort #repeat sort for a smaller list
28 done:    j after_sort

```

sort: sắp xếp mảng theo thứ tự tăng dần:

Nếu \$a1 khác \$a0, chạy nhãn max-> tìm phần tử lớn nhất của mảng chưa sắp xếp

Kết quả của nhãn max: \$v0: địa chỉ của phần tử max/ \$v1: giá trị của max

aftermax:

lw \$t0,0(\$a1): lưu phần tử cuối cùng của mảng chưa sắp xếp vào \$t0. Sau đó đưa giá trị này vào ô nhớ max.

sw \$v1,0(\$a1): lưu giá trị max vừa tính ở nhãn max vào vị trí cuối cùng của mảng chưa sắp xếp (địa chỉ lưu trong \$a1)

=> phần tử giá trị lớn nhất và phần tử cuối cùng của mảng được đổi chỗ cho nhau

addi \$a1,\$a1,-4 => tạo mảng chưa sắp xếp mới

=> lặp lại vòng lặp sort cho đến khi \$a1 = \$a0=> mảng chưa sắp xếp chỉ còn 1 phần tử A[0] thì chuyển đến nhãn done => chuyển đến after_sort

```

35 max:
36     addi $v0,$a0,0 #initialize max pointer to first el
37     lw $v1,0($v0) #initialize max value to first value
38     addi $t0,$a0,0 #initialize next pointer to first
39 loop:
40     beq $t0,$a1,ret #if next=last, return
41     addi $t0,$t0,4 #move 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

```

max: Tìm giá trị lớn nhất của mảng chưa được sắp xếp

\$v0: địa chỉ của phần tử lớn nhất được khởi tạo là địa chỉ phần tử đầu mảng.

Lưu giá trị đầu của mảng vào \$v1.

\$t0 lưu địa chỉ con trỏ

loop: Nếu con trỏ đến phần tử cuối cùng của mảng -> nhãn ret -> nhãn aftermax

Nếu không, đưa con trỏ đến ô nhớ lưu phần tử tiếp theo

Lấy giá trị của ô nhớ vị trí lưu trong \$t0 lưu vào \$t1

So sánh \$t1 với max (\$v1) (ở loop lần đầu là so sánh với A[0])

Nếu \$t1 lớn hơn max thì lưu địa chỉ phần tử vào \$v0, giá trị lưu vào \$v1. Khi đó max có giá trị mới.

\$t1 không lớn hơn max thì giữ nguyên max và lặp lại nhãn loop.

Kết thúc vòng lặp khi chuyển sang nhãn aftermax.

Final Result:

Registers			Coproc 1	Coproc 0
Name	Number	Value		
\$zero	0	0x00000000		
\$at	1	0x10010000		
\$v0	2	0x0000000a		
\$v1	3	0x00000001		
\$a0	4	0x10010000		
\$a1	5	0x10010000		
\$a2	6	0x00000000		
\$a3	7	0x00000000		
\$t0	8	0xffffffff		
\$t1	9	0xffffffff		
\$t2	10	0x00000001		
\$t3	11	0x00000000		
\$t4	12	0x00000000		
\$t5	13	0x00000000		
\$t6	14	0x00000000		
\$t7	15	0x00000000		
\$s0	16	0x00000000		
\$s1	17	0x00000000		
\$s2	18	0x00000000		
\$s3	19	0x00000000		
\$s4	20	0x00000000		
\$s5	21	0x00000000		
\$s6	22	0x00000000		
\$s7	23	0x00000000		
\$t8	24	0x00000000		
\$t9	25	0x00000000		
\$k0	26	0x00000000		
\$k1	27	0x00000000		
\$gp	28	0x10008000		
\$sp	29	0x7ffffc00		
\$fp	30	0x00000000		
\$ra	31	0x00000000		
pc		0x00400020		
hi		0x00000000		
lo		0x00000000		

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-2	1	3	5	5	5	6	6
0x10010020	7	7	8	8	59	0	0	0

3. Assignment 3

#bubble sort

#for (i=0;i<n;i++){

for (j=0;j<n-i;j++){

if (a[j]>a[j+1])

swap (a[j],a[j+1])

.data

A: .word

-5,1,3,5,7,7,-10,11,1,-5,6,7,8,10,11,60,58,7,8,
-100,-3,-9,8,31,41

Aend: .word

.text

main:

la \$a0,A

la \$a1,Aend

subi \$a1,\$a1,4

j bubble_sort

after_sort: li \$v0, 10 #exit

syscall

bubble_sort:

addi \$t0,\$a0,0

loop:

lw \$v0,0(\$t0)

lw \$v1,4(\$t0)

slt \$t1,\$v1,\$v0

beq \$t1,\$zero,skip_swap

sw \$v0,4(\$t0)

sw \$v1,0(\$t0)

skip_swap:

addi \$t0,\$t0,4

beq \$t0,\$a1,done

j loop

done: subi \$a1,\$a1,4

beq \$a1,\$a0,end_sort

j bubble_sort

end_sort: j after_sort

Result:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-100	-10	-9	-5	-5	-3	1	1
0x10010020	3	5	6	7	7	7	7	8
0x10010040	8	8	10	11	11	31	41	58
0x10010060	60	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

4. Assignment 4

#insertion_sort

.data

A: .word

-5,1,3,5,7,7,-10,11,1,-5,6,7,8,10,11,60,58,7,8
 ,-100,-3,-9,8,31,41

Aend: .word

.text

main:

la \$a0,A**la** \$a1,Aend**addi** \$t0,\$a0,4**j** insertion_sortafter_sort: **li** \$v0, 10 #exit**syscall**

end_main:

Result:

insertion_sort:

add \$v0,\$t0,0loop: **lw** \$s0,0(\$v0)**lw** \$s1,-4(\$v0)**slt** \$s2,\$s0,\$s1**beq** \$s2,\$zero,skip_swap**sw** \$s0,-4(\$v0)**sw** \$s1,(\$v0)**subi** \$v0,\$v0,4**beq** \$v0,\$a0,skip_swap**j** loop

skip_swap:

addi \$t0,\$t0,4**beq** \$t0,\$a1,end_sort**j** insertion_sortend_sort: **j** after_sort

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-100	-10	-9	-5	-5	-3	1	1
0x10010020	3	5	6	7	7	7	7	8
0x10010040	8	8	10	11	11	31	41	58
0x10010060	60	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0