Báo cáo thực hành kiến trúc máy tính giữa kỳ

Bài 1: (A - 6):

- Cách thực hiện:
  - Sử dụng hàm syscall để nhập các giá trị a, b, c từ bàn phím sau đó lưu từng giá trị vào các biến tương ứng là s0(a), s1(b), s2(c)
  - Khi nhập giá trị kiểm tra các giá trị nhập vào có dương không vì (cạnh của tam giác không thể âm hoặc = 0). Nếu âm nhảy đến nhãn warn in ra thông báo và kết thúc chương trình
  - Tạo 1 biến boolean s5 đặt là 1 (giả sử 3 cạnh là 3 cạnh của 1 tam giác)
  - Sử dụng hàm add để tính tổng a + b hoặc b+ c, c + a
  - ble để so sánh tổng 2 cạnh với cạnh còn lại nếu nhỏ hơn hoặc bằng thì nhảy đến giả sử sai
  - Giải thuật:

    if(a + b <= c || b + c <= a || a + c <= b)

    return false (Nhảy đến nhãn return_false và đặt lại giá trị của s5 = 0

  - Sử dụng hàm syscall để in giá trị s5 ra màn hình
- Soure code

20194705_HoangAnhTuan_GiuaKy_Bai1.asm

```
# Nhập 3 số nguyên a, b, c, kiểm tra đây
# có phải là 3 cạnh của một tam giác không.

.data
        warning: .asciiz "Canh cua tam giac phai lon hon 0"

.text

main:
        # Input a
        li      $v0, 5          # read integer
        syscall
        blez    $v0, warn       # if input <= 0 then warning
        nop
        add     $s0, $zero, $v0 # s0 = a

        # Input b
        li      $v0, 5          # read integer
        syscall
        blez    $v0, warn       # if input <= 0 then warning
        nop
        add     $s1, $zero, $v0 # s1 = b

        # Input c
        li      $v0, 5          # read integer
        syscall
        blez    $v0, warn       # if input <= 0 then warning
        nop
        add     $s2, $zero, $v0 # s2 = c
```

20194705_HoangAnhTuan_GiuaKy_Bai1.asm

```
31  check_triangle:
32          li      $s5, 1                  # boolean value for is triangle or not
33
34          add     $t0, $s0, $s1           # t0 = a + b
35          ble     $t0, $s2, return_false  # if a + b <= c return false
36          nop
37          # OR
38          add     $t0, $s1, $s2           # t0 = b + c
39          ble     $t0, $s0, return_false  # if b + c <= a return false
40          nop
41          # OR
42          add     $t0, $s2, $s0           # t0 = a + b
43          ble     $t0, $s1, return_false  # if a + c <= b return false
44          nop
45          j       end_main
46  return_false:
47          li      $s5, 0          # is not triangle
48          j       end_main
49  warn:
50          li      $v0, 4
51          la      $a0, warning
52          syscall
53          j       exit
54  end_main:
55          # In gia tri s5, neu la 1 thi a, b, c la 3 canh cua tam giac
56          # va nguoc lai
57          li      $v0, 1
58          add     $a0, $zero, $s5
59          syscall
60  exit:
```

- Result:

+ Trường hợp: a = 1, b, = 2, c = 3 không phải 3 cạnh của 1 tam giác nên giá trị trả về là 0

+ Trường hợp: a = 5, b, = 7, c = 10 là 3 cạnh của 1 tam giác nên giá trị trả về là 1

```
1
2
3
0
-- program is finished running (dropped off bottom) --

5
7
10
1
-- program is finished running (dropped off bottom) --
```

- Trường hợp nhập cạnh <= 0:

```
-1
Canh cua tam giac phai lon hon 0
-- program is finished running (dropped off bottom) --

1
-2
Canh cua tam giac phai lon hon 0
-- program is finished running (dropped off bottom) --
```

Bài 2: (B - 2)

- Cách thực hiện:
  - Đầu tiên nhập số phần tử của mảng -> kiểm tra hợp lệ (không âm)
  - Khởi tạo địa chỉ của mảng, lưu vào thanh ghi s5, s6
    - S6: địa chỉ mảng cố định
    - S5: con trỏ duyệt mảng
    - sử dụng stack để lưu và không làm thay đổi giá trị của thanh ghi $ra và $s0
  - Chạy vòng lặp từ 0 tới N(số phần tử mảng) để nhập từng số vào mảng
  - Hàm print để in phần tử của mảng cùng dấu phẩy
  - Hàm swap đổi giá trị của A[i] và A[j] cho nhau
  - Thực hiện 2 vòng lặp để sắp xếp số dương giảm dần
    - Nếu gặp phần tử <= 0 thì thực hiện vòng lặp tiếp theo
    - Nếu A[j] > A[i] thực hiện swap
- Source code:

*** Dòng 6 không dùng đến, em khai báo thừa ạ.

20194705_HoangAnhTuan_GiuaKy_Bai1.asm | 20194705_HoangAnhTuan_GiuaKy_Bai2.asm

```
1  # Nhập mảng số nguyên từ bàn phím.
2  # Sắp xếp các phần tử có giá trị dương giảm dần.
3
4
5  .data
6          A:      .word
7          warning: .asciiz "So phan tu cua mang khong duoc am"
8
9  .text
10 input_numbers_element:
11         li      $v0, 5
12         syscall
13         bltz    $v0, warn               # N < 0 then warning
14         nop
15         addi    $s6, $s6, 0x10010000    # s6 -> 0x10010000 = &A
16         add     $s5, $s6, $zero         # s5 -> 0x10010000 = &A
17
18         addi    $s0, $v0, 0             # s0 = N
19         li      $t1, 0                  # i = 0
20
21         sll     $s1, $s0, 2             # s1 = 4 * N
22         sub     $sp, $sp, $s1           # Khoi tao bo nho stack 4*n byte
23
24 input_element_array:
25         bge     $t1, $s0, end_input     # if (i < N)
26         nop
27         li      $v0, 5                  # read integer
28         syscall
29         sw      $v0, 0($s5)             # s5[i] = v0
30
```

20194705_HoangAnhTuan_GiuaKy_Bai1.asm | 20194705_HoangAnhTuan_GiuaKy_Bai2.asm

```
30
31          addi    $t1, $t1, 1             # i++
32          addi    $s5, $s5, 4             # s5 -> &A[i]
33
34          j       input_element_array
35  end_input:
36
37  main:
38          addi    $a0, $s6, 0             # a0 = &A[0]
39          addi    $a1, $s0, 0             # a1 = N
40          jal     sort_pos_desc
41
42          li      $t1, 0                  # i = 0
43          addi    $s5, $s6, 0             # s5 = s6 -> &A[0]
44  print:
45          bge     $t1, $s0, end_print     # if (i >= N) then exit
46          nop
47          li      $v0, 1                  # print integer
48          lw      $a0, 0($s5)             # a0 = A[i]
49          syscall
50
51          li      $v0, 11                 # print comma
52          li      $a0, 44
53          syscall
54
55          addi    $t1, $t1, 1             # i++
56          addi    $s5, $s5, 4             # s5 -> &A[i]
57
58          j       print
59  end_print:
60  end_main:
```

```
end_main:
        j       exit
swap:
        sw      $s0,0($t9)              # A[j] = A[i]
        sw      $s1,0($t8)              # A[i] = A[j]
        jr      $ra
sort_pos_desc:
        addi    $sp, $sp, -8            # Khoi tao 2 vung nho 4 byte
        sw      $ra, 4($sp)             # push $ra -> stack
        sw      $s0, 0($sp)             # push $s0 -> stack

        li      $t1, 0                  # i = 0
        li      $t2, 0                  # j = 0
        addi    $t8, $a0, 0             # v0 -> &A[0]
loop_i:
        bge     $t1, $a1, end_loop_i    # if i >= n exit loop i
        nop
        lw      $s0, 0($t8)             # s0 = A[i]
        blez    $s0, continue_i         # if (A[i] <= 0)
        nop
        addi    $t2, $t1, 1             # j = i+1
        addi    $t9, $t8, 4             # v1 -> A[i+1]
loop_j:
        bge     $t2, $a1, end_loop_j    # if (j >= n) exit loop j
        nop
        lw      $s1, 0($t9)             # s1 = A[j]
        blez    $s1, continue_j         # if (s1 <= 0) -> continue_j
        nop
        bge     $s0, $s1, continue_j    # if (A[j] > A[i]) -> swap
        nop
        jal     swap
```

```
        nop
continue_j:
        addi    $t2, $t2, 1             # j++
        addi    $t9, $t9, 4             # t9 -> &A[j]
        lw      $s0, 0($t8)             # s0 = A[i]
        j       loop_j
end_loop_j:
continue_i:
        addi    $t1, $t1, 1             # i++
        addi    $t8, $t8, 4             # t8 -> &A[i]
        j       loop_i
end_loop_i:
        lw      $s0, 0($sp)
        lw      $ra, 4($sp)
        addi    $sp, $sp, 8             # free memory
        jr      $ra
        j       exit
end_sort_pos_desc:
warn:
        li      $v0, 4
        la      $a0, warning
        syscall

exit:
```

- Result:

+ Trường hợp nhập số phần tử của mảng âm:

```
-5
So phan tu cua mang khong duoc am
-- program is finished running (dropped off bottom) --
```

+ Mảng không có phần tử nào

```
0

-- program is finished running (dropped off bottom) --
```

+ Mảng có cả phần tử âm và dương: A = [5, -4, 6, 3, -2]

Clear

```
5
5
-4
6
3
-2
6,-4,5,3,-2,
-- program is finished running (dropped off bottom) --
```

+ Mảng có phần tử dương tăng dần

Clear

```
5
1
2
3
4
5
5,4,3,2,1,
-- program is finished running (dropped off bottom) --
```

+ Mảng chỉ có phần tử âm

Clear

```
4
-4
-1
-2
-3
-4,-1,-2,-3,
-- program is finished running (dropped off bottom) --
```

Bài 3: (C – 2):

- Cách thực hiện:
    - Input chuỗi từ bàn phím:
    - Khởi tạo giá trị i và lưu địa chỉ của chuỗi và s5, count
    - Sử dụng 2 vòng lặp
        - Vòng lặp bên ngoài duyệt cả chuỗi, thoát vòng lặp bên ngoài khi gặp kí tự enter
        - Vòng lặp bên trong duyệt từ đầu tới vị trí hiện tại của i:
            - Nếu j = i tức là không có kí tự nào trước đó trùng với kí tự hiện tại và tăng count
            - So sánh kí tự hiện tại với kí tự đang xét nếu trùng thì thoát vòng lặp bên trong, không trùng thì thực hiện tiếp vòng lặp
- Source code:

20194705_HoangAnhTuan_GiuaKy_Bai1.asm | 20194705_HoangAnhTuan_GiuaKy_Bai2.asm

```
1  # Nhập vào xâu ký tự. In ra các ký tự khác nhau có trong xâu
2
3  .data
4          string: .space 100
5  .text
6  input_data:
7          li      $v0, 8
8          la      $a0, string
9          li      $a1, 100
10         syscall
11 main:
12         li      $t0, 0                  # i = 0
13         add     $s5, $zero, $a0         # s5 -> string[0]
14 getDiffChar:
15         li      $t9, 0                  # count = 0
16 outer:
17         add     $s0, $s5, $t0           # s0 = &string[i]
18         lb      $s1, 0($s0)             # s1 = string[i]
19         beq     $s1, 10, exit_getDiffChar       # is '\n' ?
20         nop
21         li      $t1, 0                  # j = 0
22
```

```
        # so sanh cac ki tu truoc do voi ki tu hien tai
        inner:
                # If i = j then count++
                beq     $t1, $t0, increase_count
                nop
                add     $s2, $s5, $t1   # s2 = &string[j]
                lb      $s3, 0($s2)     # s3 = string[j]
                beq     $s3, $s1, exit_inner    # if s2 = s3 then not increase
                nop
                addi    $t1, $t1, 1     # j++
                j       inner
        exit_inner:

        addi    $t0, $t0, 1             # i++
        j       outer
exit_outer:

increase_count:
        addi    $t9, $t9, 1     # count++
        j       exit_inner
exit_getDiffChar:

end_main:
        li      $v0, 1
        add     $a0, $zero, $t9
        syscall
```

- Result:

Clear

```
A
a
1
-- program is finished running (dropped off bottom) --

aaaabcd
4
-- program is finished running (dropped off bottom) --

hoang anh tuan
8
-- program is finished running (dropped off bottom) --
```