Báo cáo giữa kỳ

Bùi Hoàng Tú – 20200547

Bài 8 : Write a program to: You must create 1 variable (number students) and 2 arrays (student name, mark) to store the input data - Input the number of students in class. - Input the name of students in class, Math mark (0 →10) - Sort students due to their mark.

Cách thực hiện :

- Nhập vào số lượng sinh viên
- Giảm giá trị thanh ghi $sp để lưu 2 mảng tên và điểm vào stack, $s0, $s1 chứa địa chỉ đầu của 2 mảng
- Sử dụng vòng lặp để lưu điểm vào tên vào 2 mảng
- Sử dụng sắp xếp nổi bọt với con trỏ để sắp xếp sinh viên theo thứ tự từ thấp đến cao
- In ra danh sách sinh viên và điểm

Mã nguồn :

```
.data
buffer: .space 40
end_buff: .asciiz
num_of_students : .asciiz "enter number
of students : "
enter_name : .asciiz "enter your name : "
enter_score: .asciiz "enter your score : "
enter: .asciiz "\n"
.text

#save number of students
li $v0, 4
la $a0, num_of_students
syscall
li $v0,5 #read number of students
syscall
add $t0,$0,$v0 #t0 is number of students
: n
#-----------------------------------------------
# procedure memory allocate
# register:
# $t2 bytes of a unit
# $t3 amount of memory to allocate
# $s0 pointer point to head of point array
# $s1 pointer point to head of string
array
#-----------------------------------------------
```

```
C code
#include <stdio.h>
#include <stdlib.h>

/*data*/ char buffer[44];
char num_of_students[]="enter number
of students : ";
char enter_name[]= "enter your name : ";
char enter_score[]= "enter your score : ";
char enter[]= "\n";
int main (){


printf("%s",num_of_students);

int n; // $t0
scanf("%d",&n);


//malloc
int * point = malloc(n* sizeof(int));



char * name = malloc(n*40 *sizeof(char));
```

```
# array of point
li $t2,-4
mult $t0,$t2
mflo $t3 #number of stack needed for
point
add $sp,$sp,$t3                          int *mpoint = point;
add $s0,$sp,$0 #head of point array      char *mname = name;
#array of name
li $t2,-40
mult $t0,$t2
mflo $t3 #number of stack needed for
name
add $sp,$sp,$t3
add $s1,$sp,$0 #head of name array
#-------------------------------------------------
# procedure input, loop n times           //input loop
# register:                               for(int i = 0;i<n;i++){
# $t2 index of students
# $s2 pointer point to array
# $s3 pointer point to string
#-------------------------------------------------
li $t2,0                                 printf("%s",enter_name);
add $s2,$0,$s0 #move pointer of point
add $s3,$0,$s1 #move pointer of string
Loop:
addi $t2,$t2,1
bgt $t2,$t0,end_inp #for i : 1->20,      scanf("%s",mname);
t2:index
nop

#print enter line
li $v0, 4
la $a0, enter_name
syscall
li $v0,8 #take in input
add $a0, $s3,$0 #load byte space into
array pointer
li $a1, 40 # max length
syscall
#-----------------------------------------------------
# procedure check input point             // input check
# registers:                              do{
# $t1 : register check for entered point > printf("%s",enter_score);
10
# $t3 : register check for entered point <
0
# $t4 : register check for entered point <  scanf("%d",mpoint);
0 or >10                                 }while(*mpoint <0||*mpoint>10);
```

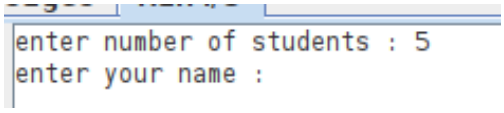
```
#-----------------------------------------------------
re_score:
li $v0,4
la $a0, enter_score
syscall
li $v0,5 #read int
syscall                                  //move pointer
sgt $t1,$v0,10                           mname +=40;
sgt $t3,$0,$v0                           mpoint +=1;
add $t4,$t1,$t3                          }
bne $t4,$0, re_score
nop
#if pass both, save to point array
sw $v0, 0($s2)
#move pointer, index
addi $s3,$s3,40
addi $s2,$s2,4
j Loop #for i : 1->n
nop
end_inp:
#sort from here                           //sort using pointer
#-----------------------------------------------------  char tmpc;
# procedure sort (descending selection    int tmpi1,tmpi2;
sort using pointer)                      int *mpoint = point;
# register:                              char *mname = name;
# $t1 index of students                  for (int i=0;i<n;i++){
# $s2 pointer point to array
# $s3 pointer point to string            tmpi1 = *(mpoint+i);
# $t3 min of unsorted part
# $t6 current point
# $t8,$t7 tmp for switching
#-----------------------------------------------------
li $t1,0                                 for (int j=i+1;j<n;j++){
add $s2,$0,$s0 #move pointer of point    tmpi2= *(mpoint+j);
add $s3,$0,$s1 #move pointer of string
sort:
beq $t1,$t0,done_sort
lw $t3,0($s2) #min = point[0]
add $t2,$t1,$0
# stack move
add $t5,$s2,$0
add $t4,$s3,$0
bb_loop:
#a[j] move
addi $t5,$t5,4
addi $t4,$t4,40
addi $t2,$t2,1
beq $t2,$t0,done_bb
```

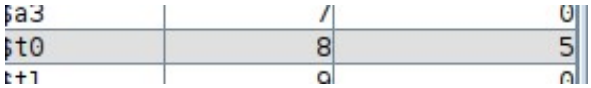
```
lw $t6,0($t5) #this (a[i])
slt $t7, $t6,$t3
bne $t7,$0,bb_loop
## switch                                  //switch
#switch point                             if (tmpi1<tmpi2){
add $t8,$t6,$0                            //switch point;
add $t6,$t3,$0                            *(mpoint+i)=tmpi2;
add $t3,$t8,$0                            *(mpoint+j)=tmpi1;
sw $t3,0($s2)                             tmpi1 = *(mpoint+i);
sw $t6,0($t5)                             tmpi2= *(mpoint+j);
li $s6,0
#switch name                              //switch name
switchloop:                               for (int k=0;k<40;k++){
add $s5,$s3,$s6 #index 1                  tmpc= *(mname+i+k);
add $s4,$t4,$s6 #index 2                  *(mname+i+k) =*(mname+j+k);
lw $t7,0($s5) #t7 to change first         *(mname+j+k) = tmpc;
lw $t8,0($s4) #t8 buffer                  }
sw $t7,0($s4)                             }
sw $t8 0($s5)                             }
addi $s6,$s6,4                            }
beq $s6,40,switch_out
nop
j switchloop
nop
switch_out:
done_bb:
addi $t1,$t1,1
addi $s2,$s2,4 #next position on stack
addi $s3,$s3,40
j sort
done_sort:
#---------------------------------------------  //print
# procedure print, loop n times            int *mpoint = point;
# register:                                char *mname = name;
# $t1 index of students                    for (int i=0;i<n;i++){
# $t4 pointer point to array               printf("%s%d",mname+i*40,*(mpoint+i));
# $t5 pointer point to string
#---------------------------------------------  }
li $t1,0                                   }
add $t4,$0,$s0 #to top point
add $t5,$0,$s1 #top string
print_all:
beq $t1,$t0,done_print
nop
#print_name
li $v0,4
add $a0,$0,$t5
syscall
```

```
li $v0,1
lw $a0, 0($t4)
syscall
li $v0,4
la $a0,enter
syscall
addi $t1,$t1,1
addi $t4,$t4,4
addi $t5,$t5,40
j print_all
nop
done_print:
```

I, Trong procedure « nhập số lượng sinh viên »

```
#save number of students
li $v0, 4
la $a0, num_of_students
syscall
li $v0,5 #read number of students
syscall
add $t0,$0,$v0 #t0 is number of students : n
```





Thanh ghi t0 lưu biến toàn cục : số lượng sinh viên là 5

II, Procedure « khai báo 2 mảng điểm và tên của sinh viên »

```
li $t2,-4
mult $t0,$t2
mflo $t3 #number of stack needed for point
add $sp,$sp,$t3
add $s0,$sp,$0 #head of point array
#array of name
li $t2,-40
mult $t0,$t2
mflo $t3 #number of stack needed for name
add $sp,$sp,$t3
add $s1,$sp,$0 #head of name array
```

Thanh ghi t3 lưu lượng bytes cần dùng cho mỗi mảng

Thanh ghi sp tăng lên để tạo chỗ cho mảng số

$t3	11	-20
$sp	29	0x7fffefe8

Thanh ghi sp tăng lên để tạo chỗ cho mảng tên

hi		-1
lo		-200

$t3	11	-200
$sp	29	0x7fffef20

Thanh ghi s0 và s1 lưu biến toàn cục con trỏ đến mảng điểm và mảng tên

III, Procedure input

Dùng 2 thanh ghi s2, s3 trỏ đến từng phần tử của 2 mảng để lưu nhập vào và thanh ghi t2 lưu số lượng đã nhập, khi số lượng nhập vượt quá số sinh viên sẽ thoát khỏi vòng lặp

1, Vòng lặp đầu tiên

$s2	18	0x7fffefe8
$s3	19	0x7fffef20
$t2	10	0x00000001

s2,s3 đang trỏ đến phần tử đầu tiên 2 mảng.

$t2 = 1, chưa lớn hơn 20 nên chương trình tiếp tục

Chương trình dùng syscall 8 để nhập tên

$a0	4	0x7fffef20
$a1	5	0x00000028

a0 = s3 (phần tử đầu của mảng tên)

a1 = $28_{16} = 40_{10}$ là số lượng phần tử tối đa

Nhập vào : `enter your name : Bui Hoang Tu 20200547`

Dữ liệu trong mảng như sau :

i	u	B		n	a	o	H		u	T		g	2	0	2		4	5	0	0	\0	\0	\n	7		\0	\0	\0	\0			

Chương trình sau đó yêu cầu nhập điểm, nếu nhập sai khoảng (<0 hoặc >10) chương trình yêu cầu nhập lại. Sử dụng thanh ghi t1 kiểm tra >10 và thanh ghi t3 kiểm tra < 1, thanh ghi t4 tổng hợp 2 giá trị trên. Với đầu vào <0 sau

```
enter your score : -1
```

Giá trị các thanh ghi là :

$t1	9	0x00000000
$t2	10	0x00000001
$t3	11	0x00000001
$t4	12	0x00000001

Chương trình yêu cầu nhập lại điểm :

```
enter your score : -1
enter your score : 11
```

Với đầu vào 11 >10 , chương trình tiếp tục yêu cầu nhập lại điểm:

$t1	9	0x00000001
$t2	10	0x00000001
$t3	11	0x00000000
$t4	12	0x00000001

```
enter your score : -1
enter your score : 11
enter your score : 10
```

Với đầu vào đúng, chương trình tiếp tục, các thanh ghi và mảng như sau :

$t1	9	0x00000000
$t2	10	0x00000001
$t3	11	0x00000000
$t4	12	0x00000000

10	0	0

Chương trình lặp lại tương tự đến khi số sinh viên nhập vào = số lượng nhập vào ban đầu

IV, procedure sort sắp xếp với sử dụng các con trỏ.

Tại mỗi phần tử, tìm phần tử lớn nhất của phần còn lại và hoán đổi vị trí cho nhau.

$t1 chỉ số
$s2 con trỏ đến điểm hiện tại (cần tìm max)
$s3 con trỏ đến tên hiện tại
$t3 giá trị của điểm hiện tại
$t6 giá trị của các phần tử phía sau hiện tại
$t8,$t7 các thanh ghi lưu giá trị tạm để đổi vị trí

Xét vòng lặp thứ 2 :

$t1	9	1	Chỉ số hiện tại:1
$t2	10	2	Chỉ số tạm thời:2
$t3	11	5	Giá trị hiện tại : 5
$t4	12	2147479408	Giá trị tạm thời:8
$t5	13	2147479536	
$t6	14	8	

Giá trị hiện tại nhỏ hơn nên đảo vị trí 2 phần tử

$t1	9	1
$t2	10	2
$t3	11	8
$t4	12	2147479408
$t5	13	2147479536
$t6	14	5

$t1	9	1	Chỉ số hiện tại:1
$t2	10	3	Chỉ số tạm thời:3
$t3	11	8	Giá trị hiện tại : 8
$t4	12	2147479448	Giá trị tạm thời:4
$t5	13	2147479540	
$t6	14	4	

Giá trị hiện tại lớn hơn nên không đảo vị trí 2 phần tử

$t1	9	1	Chỉ số hiện tại:1
$t2	10	4	Chỉ số tạm thời:4
$t3	11	8	Giá trị hiện tại : 8
$t4	12	2147479488	Giá trị tạm thời:9
$t5	13	2147479544	
$t6	14	9	

Giá trị hiện tại nhỏ hơn nên đảo vị trí 2 phần tử

$t1	9	1
$t2	10	4
$t3	11	9
$t4	12	2147479488
$t5	13	2147479544
$t6	14	8

Vòng lặp đầu tiên và các vòng lặp sau, các thanh ghi có hành vi tương tự.

V, In ra kết quả

Đầu vào và kết quả sau khi sắp xếp như sau :

```
enter number of students : 5
enter your name : Bui Hoang Tu 20200547
enter your score : 10
enter your name : name 2
enter your score : 5
enter your name : name 3
enter your score : 8
enter your name : name 4
enter your score : 4
enter your name : name 5
enter your score : 9
Bui Hoang Tu 20200547
10
name 5
9
name 3
8
name 2
5
name 4
4
```

Bài 9 : Write a program to: Assume that you already have 1 variable (number students) and 2 arrays (student name, mark) in memory - Read in the number of students in the class. - Read information about each student, including: Name, Math mark. - List the names of all students who have not passed the Math exam.

Cách thực hiện :

- B1 : Nhập giá trị của “Điểm số tối thiểu để đạt” và lưu biến
- B2 : Đọc số lượng sinh viên và danh sách tên và điểm của các học sinh
- B3 : Sử dụng vòng lặp để thay phiên kiểm tra các giá trị cần thiết
- B4 : In ra danh sách sinh viên không đạt môn Toán

Mã nguồn :

```
# Assembly code

.data
Number_of_student : .word 3
Name:          .asciiz"Mai Nguyen Ngoc
Huy", "Bui Hoang Tu 2020054","name3
MSSV"
Score:.word 5, 9, 1
text: .asciiz " Minimum score required : "
text1: .asciiz " \n List the names of all
students who have not passed the Math
exam : \n\n"
blank: .asciiz " "
newline: .asciiz "\n"
dot: .asciiz ". "
.text
#------------------------------------------------------
# Procedure to save the number of
students and the Minimum Score required
to pass the exam
# just work with interger number only
# Register :
# $t0 : The number of student
# $s0 : Pointer point to head of string
array
# $s1 : Pointer point to head of score
array
# $t5 : Minimum score required to pass
the exam
# $s3, $s4, $s5 : Temporary variable to
check the input condition of minimum
score required
```

```
#include <stdio.h>

// .Data
#define NUM_OF_STUDENTS 3
char Name[][40] = {"Mai Nguyen Ngoc
Huy", "Bui Hoang Tu 2020054","name3
MSSV"};
int Score[] = {5,9,1};




int main() {
// Procedure to save the Minimum Score
required to pass the exam
printf("Minimum Score Require : ");
int m;
scanf("%d", &m);
int n = NUM_OF_STUDENTS;
```

```
#-------------------------------------------------------
minimum_score :
li $v0, 4
la $a0, text
syscall
li $v0, 5 # read int input
syscall
add $t5, $0, $v0 # Save the value just
entered into the register $s5
addi $s3, $0, 11 #
slt $s4, $t5, $s3 # check the student
score condition
addi $s3, $0, -1 # ( Between from 0 and
10 )
slt $s5, $s3, $t5 #
bne $s5, $s4, minimum_score #
# Save the number of student and set
pointers to the head of the string array
and the score array
lw $t0, Number_of_student
la $s0, Name
la $s1, Score
li $v0, 4
la $a0, text1
syscall
#-------------------------------------------------------
# Procedure to compare each student's
score with the minimum score required to
print the student's
# information with 2 nested loops
# ------
# Register :
# $t0 : The number of student
# $s0 : Pointer point to head of string
array
# $s1 : Pointer point to head of score
array
# $t5 : Minimum score required to pass
the exam
# $t1 : index of first loop
# $t2 : pointer point to score array
# $t3 : index for each string of string
array in the second loop
# $t6 : Temporary variable to save the
value of the compare between student's
score and minimum
# score required
# $a1 : Variable to print order number
```

```
printf("\n List the names of all students
who have not passed the Math exam : \n
");

// compare points
for(int i = 0; i < n; i++) {
if (Score[i] < m) {
```

```
#
-------------------------------------------------------
-------------------------
name_loop:
beq $t1, $t0, exit
lw $t2, 0 ($s1)
slt $t6, $t2, $t5 # Compare the student's
score with the minimum
# score required
# Procedure print the ordinal number
beq $t6, $0, print_name
add $a1, $a1, 1
li $v0, 4 # print a blank
la $a0, blank
syscall
li $v0, 1 # print the ordinal number
add $a0, $0, $a1
syscall
li $v0, 4
la $a0, dot
syscall
#-------------------------------------------------------
# Procedure to print student's name by
printing each letter if the student's score
meets the
# requirements of the question paper
#-------------------------------------------------------
print_name:
li $v0, 11
lb $a0, 0($s0) # load the current $s0's
input bytes
add $t3, $0, $a0 # save $s0 into $t3
beq $t6, $0, cancel
syscall
cancel:
add $s0, $s0, 1 # increase the byte of
current string by 1
bne $t3, $0, print_name # if the byte not
zero, continue loop
nop
beq $t6, $0, skip
li $v0, 4 # print a blank
la $a0, blank
syscall
li $v0, 1 # read number
lw $a0, 0 ($s1) # load the score to $a0
and print
syscall #
```

```
// print name
printf(" %d. %s %d", i+1, Name[i],
Score[i]);
}
}
}
```

```
li $v0, 4 # print a new line characters
la $a0, newline
syscall
skip: add $s1, $s1, 4
add $t1, $t1, 1
j name_loop
exit:
```

I, Procedure input " Minimum Score Require "

$t0 : The number of student

$s0 : Pointer point to head of string array

$s1 : Pointer point to head of score array

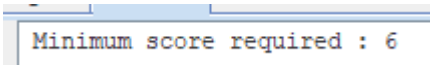
$t5 : Minimum score required to pass the exam

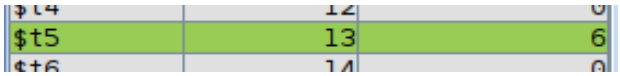
$s3, $s4, $s5 : Temporary variable to check the input condition of minimum score required

 # Take input Minimum score require

```
li      $v0,  4
la              $a0,  text
syscall

li      $v0,  5
syscall
add   $t5,   $0,    $v0
```





- Giá trị đầu vào vừa nhập được lưu vào thanh ghi $t5 lưu biến toàn cục : Điểm số yêu cầu để qua môn là 6

 # Check the input condition

```
addi  $s3,  $0,    11
slt     $s4,  $t5,   $s3
addi  $s3,  $0,    -1
slt     $s5,  $s3,  $t5
bne   $s5,  $s4,  minimum_score
```

$s3	19	-1
$s4	20	1
$s5	21	1

- Dùng các giá trị tương đối là 11 và -1 để giới hạn giá trị điểm số đầu vào ($t5 >= 0 và $t5 <= 10)

II, Procedure "Thiết lập các giá trị cơ bản"

```
lw      $t0,  Number_of_student
la      $s0,  Name
la      $s1,  Score
```

Thiết lập các con trỏ trỏ đến các đại lượng cần thiết

III, Procedure Thực hiện yêu cầu đề bài bằng vòng lặp duyệt qua tất cả sinh viên và vòng lặp duyệt đến tên sinh viên tiếp theo. Sử dụng các con trỏ đã thiết lập ở Phần II rồi in ra màn hình

Tại mỗi phần tử có số thứ tự tương ứng với mảng "Tên học sinh" và "Điểm thi Toán" :

- So sánh điểm số của học sinh số thứ tự hiện tại với điểm số yêu cầu, nếu thỏa mãn điều kiện (thấp hơn điểm qua môn), thực hiện :
  - Bên cạnh đó sử dụng thêm 1 hàm in thông tin của những sinh viên trượt môn Toán.
  - In thông tin

- Xem xét Thực hiện vòng lặp 1 (y = $t2)

$t1	9	0
$t2	10	5

Điểm toán của học sinh stt 1 lưu ở $t2

```
slt   $t6,  $t2,  $t5 #câu lệnh so sánh
beq   $t6,  $0,   print_name
```

$t2	10	5
$t5	13	6
$t6	14	1

- Lúc này, Điểm hiện tại là 5 < 6 (Điểm yêu cầu), giá tri $t6 = 1, thỏa mãn điều kiện và nhảy đến phần in.

```
print_name:
        li      $v0,    11
        lb      $a0,    0($s0)              # load the current $s0's input bytes
        add     $t3,    $0,     $a0         # save $s0 into $t3

        beq     $t6,    $0,     cancel
        syscall

        cancel:
                add     $s0,    $s0,    1       # increase the byte of current string by 1
                bne     $t3,    $0,     print_name    # if the byte not zero, continue loop
                nop
```

Phần này được thực hiện bằng cách sử dụng con trỏ vào đầu các Mảng Tên và Mảng điểm Toán, đồng thời load byte để thực hiện vòng lặp. Thực hiện in chuỗi và nhảy con trỏ từng kí tự cho đến qua ký tự kết thúc chuỗi 1 byte (đến chuỗi tiếp theo) .

- Xem xét lần lặp thứ 2

$t2	10	9
$t5	13	6
$t6	14	0

  - Điểm số hiện tại là ($t2 == 9) so sánh với điều kiện ($t5 == 6) lớn hơn nên $t6 == 0. Chương trình bỏ qua phần in.
- Tương tự với lần lặp thứ 3

IV, Kết quả đầu ra

```
Minimum score required : 6

List the names of all students who have not passed the Math exam :

1. Mai Nguyen Ngoc Huy 5
2. name3 MSSV 1

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