HO CHI MINH UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

COMPUTER ARCHITECTURE

Assignment - Semester 1 - 2020

Lecturers: Assoc. Prof. Dr. Cuong Pham Quoc

Mr. Kieu Do Nguyen Binh

Class: CC04 Group: 8

Members: Nguyễn Minh Hùng Student ID: 1952737

Phạm Nhật Hoàng Student ID: 1952703

Nguyễn Bá Minh Hưng Student ID: 1952748

Question 1. Given the following MIPS declaration in the data section of a MIPS program

```
.data
 nums .word <an integer number>
 elems .word <array elements>
```

Where <an integer number> will store the number of elements in the array elems. elems is an array that stores integer elements whose size is equal to value <an integer number>. You are required to choose those values when developing and testing your program

1. Write a MIPS program that sort the the array elems in ascending order using the bubble sort algorithm. (2 points)

2. Calculate the execution time of your program if one instruction requires 1 ns for processing. (1 point)

Answer:

1.

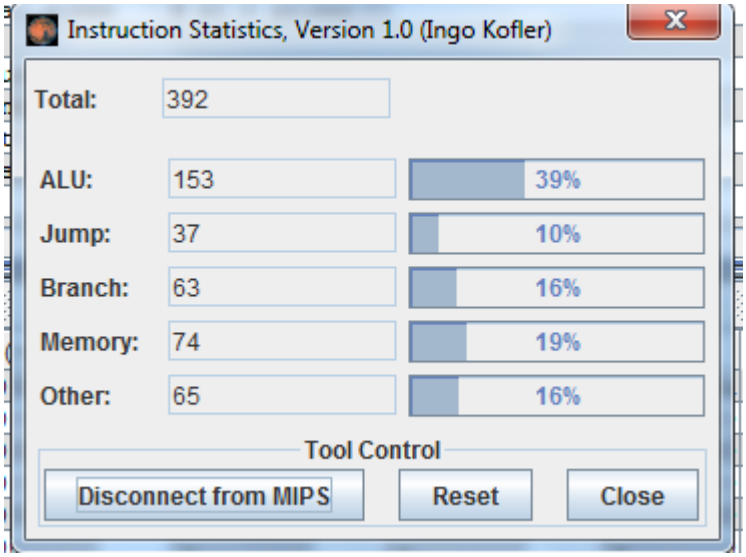
```
.data
nums:           .word 7
elems:          .word 5,6,12,1,3,10,2
tab:            .asciiz "  "

.text
.globl main

main:
la              $s1,nums
lw              $a0,0($s1)
la              $a1,elems       # a1 = &elems[0]
# a0 <=> N
# a1 <=> address of elems
jal             bubble_sort

j               end
#-----------------------------------------------
bubble_sort:
#  t0  <=>  i
#  s0  <=> N - 1
#  t1  <=> j
#  s1  <=> N - i - 1
#  t3  <=> 4*j
#  a2  <=> address of arr[j]
#  s4  <=> arr[j]
#  s5  <=> arr[j+1]
#  t4  <=> (arr[j+1] < arr[j]) ?

li              $t0,0           # int i = 0
subi            $s0,$a0,1       # s0 = N - 1
loop_i:                         # for(int i = 0; i < N - 1;i++)
```

```
33 beq             $t0,$s0,exit_i
34 li              $t1,0          # int j = 0
35 sub             $s1,$s0,$t0    # N - i - 1
36 loop_j:                        # for(int j = 0;j < N - i -1;j++)
37 beq             $t1,$s1,exit_j
38 mul             $t3,$t1,4
39 add             $a2,$a1,$t3    # &arr[j]
40 lw              $s4,0($a2)     #  arr[j]
41 lw              $s5,4($a2)     # arr[j+1]
42 slt             $t4,$s5,$s4
43 beq             $t4,0,next
44 # swap
45 sw              $s5,0($a2)
46 sw              $s4,4($a2)
47 next:
48 addi            $t1,$t1,1      # j++
49 j               loop_j
50 exit_j:
51 addi            $t0,$t0,1      # i++
52 j               loop_i
53 exit_i:
54
55 jr              $ra
56 #-------------------------------------------------
57 end:
58 move            $t6,$a0
59 li              $t0,0
60 # print section
61 loop:
62 beq             $t0,$t6,exit
63 lw              $t1,0($a1)
64 li              $v0,1
65 move            $a0,$t1
66                  syscall
67 li              $v0,4
68 la              $a0,tab
69                  syscall
70 addi            $a1,$a1,4
71 addi            $t0,$t0,1
72 j               loop
73 exit:
74
75
76 li              $v0,10
77                  syscall
```

bubblesort.asm

2.

When we assembled our whole program (including print section), we received Instruction Statistics below:



As it can be seen from the picture, the program runs the bubble_sort algorithm with 37 jump instructions (including about 27 jump instructions for the nested - loop, which is exactly n(n-1)/2 and n - 1 times jumping out according to bubble_sort (n = 7), the others are jal, jr and loop in print section)

ALU and memory ins took up lots because we had to make a lot of comparisons in the algorithm.

If 1 ins. requires 1 ns for processing, the total instructions are 392
=> **Execution time = 1 x 392 = 392 ns.**

Question 2.

```
struct Students {
 unsigned int id;
 char name[25];
 unsigned int age:7;
 unsigned int is_male:1;
 float average_score;
} student[5];

void print_student(int std_idx){
 char genre[] = "Male\0|Female";
 printf("Student id: %u\n", student[std_idx].id);
 printf("Student name: %s\n", student[std_idx].name);
 printf("Student age: %u\n", student[std_idx].age);
 printf("Student gender: %s\n", &genre[student[std_idx].is_male==1? 0: 6]);
 printf("Student id: %f\n\n", student[std_idx].average_score);
}

void main(int n) {
 int i;
 printf("This is a list of students\n");
 /* Assign information for list of students */
 for (i = 0; i < 5; i = i + 1) print_student(i);
}
```

You are required to finish the following requirements:

1. Organize memory allocation for the **Students struct** (padding is required). (2 points)

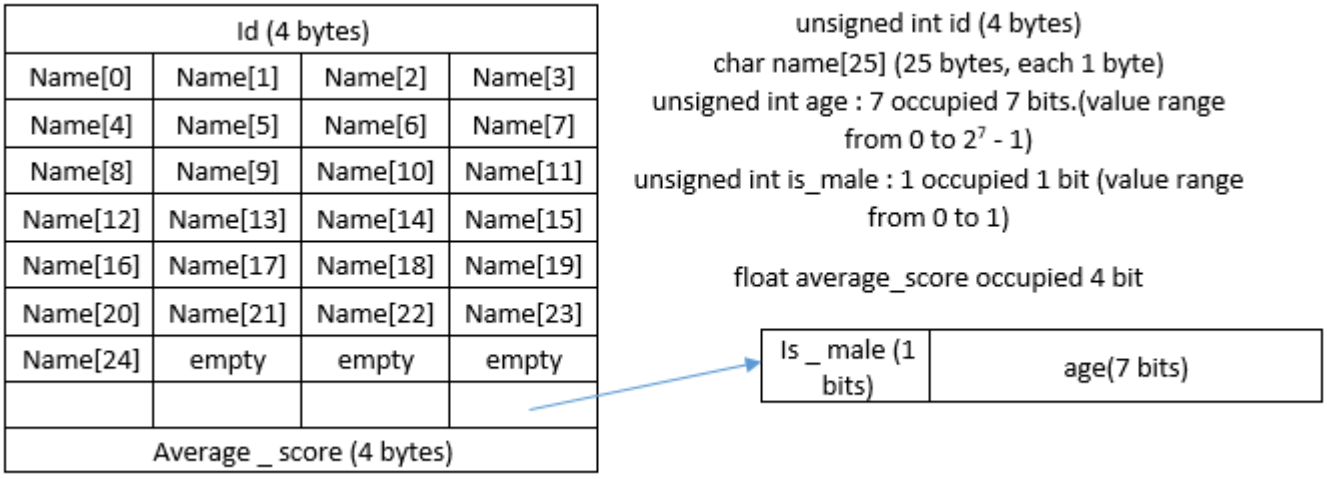
2. Initialize the array of 5 students. Assign any value for their information on the main program. (Be careful that the assigned value do not exceed the range of variables). (1 points)

3. Write a MIPS program for print student procedure. (2 points)

4. Use print student procedure to print information of 5 assigned students. (2 points)

Answer:

1. (each cell in the table is equivalent to 1 byte)



According to the table, we can conclude that struct students size is 40 bytes.

2. The following C code is initialization of the array of 5 students.

```
/* Assign information for list of students */
    student[0].id = 1952737;
    strcpy_s(student[0].name, "Minh Hung");
    student[0].age = 20;
    student[0].is_male = 1;
    student[0].average_score = 8.5;

    student[1].id = 1952703;
    strcpy_s(student[0].name, "Nhat Hoang");
    student[1].age = 19;
    student[1].is_male = 1;
    student[1].average_score = 8.1;

    student[2].id = 1952777;
    strcpy_s(student[0].name, "Van A");
    student[2].age = 22;
    student[2].is_male = 0;
    student[2].average_score = 9;
```

```
19
20      student[3].id = 1952743;
21      strcpy_s(student[0].name, "Nguyen Hung");
22      student[3].age = 18;
23      student[3].is_male = 0;
24      student[3].average_score = 7.5;
25
26      student[4].id = 1950000;
27      strcpy_s(student[0].name, "John");
28      student[4].age = 20;
29      student[4].is_male = 1;
30      student[4].average_score = 4.5;
```

Listing 1: Initialization part

3. print_student function in MIPS:

```
1  # the following is a small part of full program to  describe print_student
       procedure
2
3  print_student:                          # print_student procedure
4  mul                 $s4,$a1,40          # $s0 --> address of student[], $a1 -->
       i,
5  add                 $s5,$s0,$s4
6  lw                  $s3,0($s5)
7  li                  $v0,4
8  la                  $a0,id
9                      syscall
10 li                  $v0,1
11 move                $a0,$s3
12                     syscall
13 li                  $v0,4
14 la                  $a0,name
15                     syscall
16 lw                  $s3,4($s5)
17 li                  $s4,0
18 loop_name:                           # loop to load each byte to get a full
       name from char arr
19 beq                 $s4,11,exit_name
20 li                  $v0,11
21 lb                  $a0,0($s3)
22                     syscall
23 addi                $s3,$s3,1
24 addi                $s4,$s4,1
25 j                   loop_name
26 exit_name:
27 li                  $v0,4
28 la                  $a0,age             # load and print age information
29                     syscall
30 lb                  $s3,35($s5)
31 li                  $v0,1
32 move                $a0,$s3
33                     syscall
34 li                  $v0,4
35 la                  $a0,is_male         # load and print is_male information
36                     syscall
37 lb                  $s3,34($s5)
38 beq                 $s3,1,male          # if (is_male == 1) print "Male" else "
       Female"
39 li                  $v0,4
```

```
la                  $a0,Female
                    syscall
j                   out
male:
li                  $v0,4
la                  $a0,Male
                    syscall
out:
li                  $v0,4
la                  $a0,average          # load and print average score
                    syscall
l.s                 $f0,36($s5)
li                  $v0,2
mov.s               $f12,$f0
                    syscall
li                  $v0,4
la                  $a0,endline
                    syscall
jr                  $ra
```

print_student.asm

4. The full MIPS code program including:
initialization of student array, print_procedure, and printing information of 5 assigned students:

```
.data
student:            .space  200         # 40 bytes x 5
hung:               .byte   'M','i','n','h',' ','H','u','n','g',' ',' '
hoang:              .byte   'N','h','a','t',' ','H','o','a','n','g',' '
van_a:              .byte   'V','a','n',' ','A',' ',' ',' ',' ',' ',' ',' '
nguyen_hung:        .byte   'N','g','u','y','e','n',' ','H','u','n','g'
john:               .byte   'J','o','h','n',' ',' ',' ',' ',' ',' ',' ',' '
msg:                .asciiz "This is a list of students\n"
id:                 .asciiz "Student id: "
name:               .asciiz "\nStudent name: "
age:                .asciiz "\nStudent age: "
is_male:            .asciiz "\nStudent gender: "
Male:               .asciiz "Male"
Female:             .asciiz "Female"
average:            .asciiz "\nStudent score: "
endline:            .asciiz "\n\n"
grade1:             .float  8.5
grade2:             .float  8.1
grade3:             .float  9.0
grade4:             .float  7.5
grade5:             .float  4.5
.text
.globl main

main:

li                  $a1,0               # int i = 0
li                  $v0,4
la                  $a0,msg             # " This is ... "
                    syscall

# initialization section

la                  $s0,student
```

```
35 la                   $t0,hung
36 la                   $t1,hoang
37 la                   $t2,van_a
38 la                   $t3,nguyen_hung
39 la                   $t4,john
40
41 # student[0]
42 li                   $t5,1952737              # std id
43 sw                   $t5,0($s0)
44 sw                   $t0,4($s0)
45 li                   $t6,1                    # is_male
46 sb                   $t6,34($s0)
47 li                   $t6,20                   # age
48 sb                   $t6,35($s0)
49 l.s                  $f0,grade1               # grade
50 s.s                  $f0,36($s0)
51
52 # student[1]
53 li                   $t5,1952703
54 sw                   $t5,40($s0)
55 sw                   $t1,44($s0)
56 li                   $t6,1
57 sb                   $t6,74($s0)
58 li                   $t6,19
59 sb                   $t6,75($s0)
60 l.s                  $f0,grade2
61 s.s                  $f0,76($s0)
62
63 # student[2]
64 li                   $t5,1952777
65 sw                   $t5,80($s0)
66 sw                   $t2,84($s0)
67 li                   $t6,0
68 sb                   $t6,114($s0)
69 li                   $t6,22
70 sb                   $t6,115($s0)
71 l.s                  $f0,grade3
72 s.s                  $f0,116($s0)
73
74 # student[3]
75 li                   $t5,1952743
76 sw                   $t5,120($s0)
77 sw                   $t3,124($s0)
78 li                   $t6,0
79 sb                   $t6,154($s0)
80 li                   $t6,18
81 sb                   $t6,155($s0)
82 l.s                  $f0,grade4
83 s.s                  $f0,156($s0)
84
85 # student[4]
86 li                   $t5,1950000
87 sw                   $t5,160($s0)
88 sw                   $t4,164($s0)
89 li                   $t6,1
90 sb                   $t6,194($s0)
91 li                   $t6,20
92 sb                   $t6,195($s0)
93 l.s                  $f0,grade5
94 s.s                  $f0,196($s0)
```

```
95
96
97  print_loop:
98  beq                 $a1,5,exit_loop
99  jal                 print_student
100 addi                $a1,$a1,1
101 j                   print_loop
102
103 print_student:                           # print_student procedure
104 mul                 $s4,$a1,40          # $s0 --> address of student[], $a1 -->
        i,
105 add                 $s5,$s0,$s4
106 lw                  $s3,0($s5)
107 li                  $v0,4
108 la                  $a0,id
109                     syscall
110 li                  $v0,1
111 move                $a0,$s3
112                     syscall
113 li                  $v0,4
114 la                  $a0,name
115                     syscall
116 lw                  $s3,4($s5)
117 li                  $s4,0
118 loop_name:                            # loop to load each byte to get a full
        name from char arr
119 beq                 $s4,11,exit_name
120 li                  $v0,11
121 lb                  $a0,0($s3)
122                     syscall
123 addi                $s3,$s3,1
124 addi                $s4,$s4,1
125 j                   loop_name
126 exit_name:
127 li                  $v0,4
128 la                  $a0,age             # load and print age information
129                     syscall
130 lb                  $s3,35($s5)
131 li                  $v0,1
132 move                $a0,$s3
133                     syscall
134 li                  $v0,4
135 la                  $a0,is_male         # load and print is_male information
136                     syscall
137 lb                  $s3,34($s5)
138 beq                 $s3,1,male          # if (is_male == 1) print "Male" else "
        Female"
139 li                  $v0,4
140 la                  $a0,Female
141                     syscall
142 j                   out
143 male:
144 li                  $v0,4
145 la                  $a0,Male
146                     syscall
147 out:
148 li                  $v0,4
149 la                  $a0,average         # load and print average score
150                     syscall
151 l.s                 $f0,36($s5)
```

```
152 li                  $v0,2
153 mov.s               $f12,$f0
154                     syscall
155 li                  $v0,4
156 la                  $a0,endline
157                     syscall
158 jr                  $ra
159
160 exit_loop:
161
162 end:
```

struct.asm

< all source codes are included in code folder >

——————the end——————