COMP1521
WEEK 4 – MIPS function

Siyu (Annie) Qiu

1. Mips 2D-array
2. Structs
3. Mips function
4. Style for ass!!!

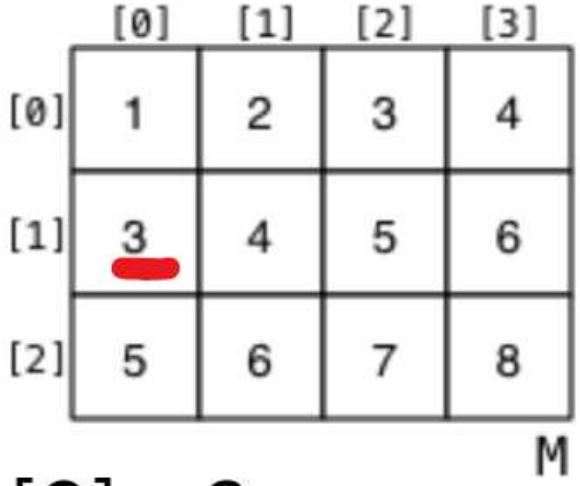
Announcements 

- Lab start this week!
  - Lab4 has been released, due on **Week 5 Monday 12:00:00 (midday)**

- weekly quiz will be **due Week 5 Thursday 21:00:00**
- **Assignment1** will due on Week 5 Friday 18:00:00

Multi-Dimentional Arrays

consider a 3 x 4 array:



M[1][0] = 3

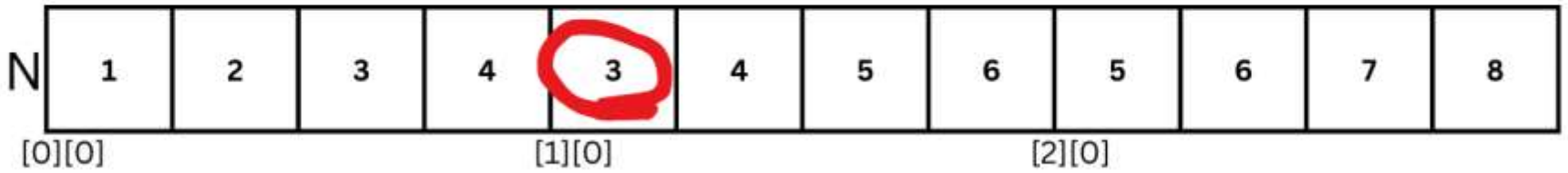
consider a 3 x 4 array:

	[0]	[1]	[2]	[3]
[0]	1	2	3	4
[1]	3	4	5	6
[2]	5	6	7	8

M

M[1][0] = 3

In memory this 3 x 4 array would look like:

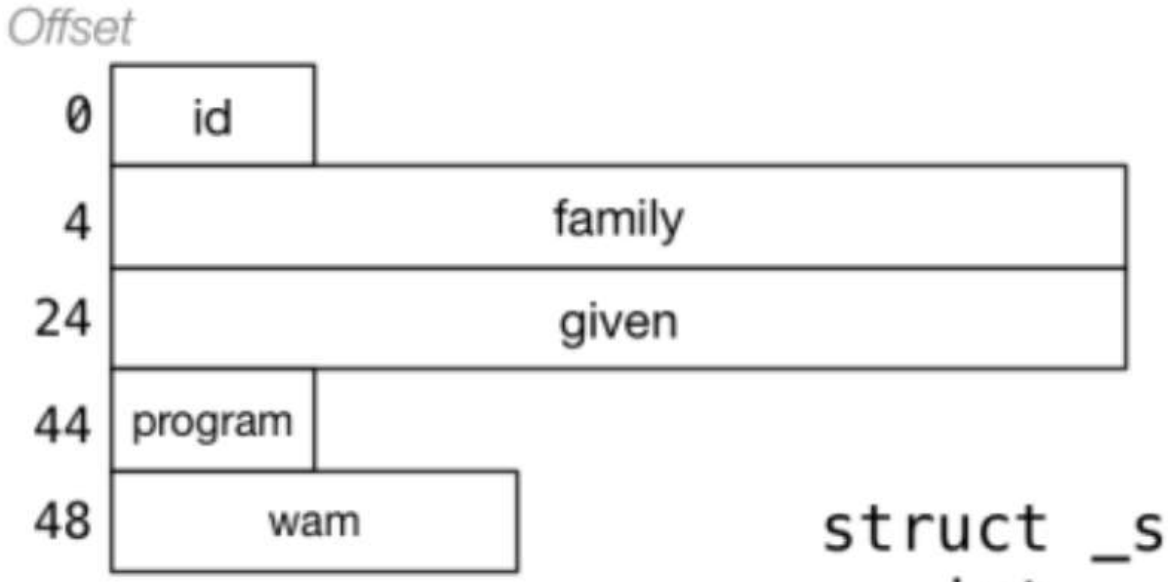


Summary

- Total_offset = rows * num_cols + cols
- multiply total_offset by a constant to adjust for element size
  - Int / char
  - Tutorial question 2

Structs

Structs in MIPS




```
struct _student {
    int     id;
    char    family[20];
    char    given[20];
    int     program;
    double wam;
};
```

usually good code (including those given in labs/assignments) will have the offset calculated for you previously and will be a #DEFINE.

```
ID_OFFSET = 0
FAMILY_OFFSET = 4
GIVEN_OFFSET = 24
PROGRAM_OFFSET = 44
WAM_OFFSET = 48

# to access the program,
main:
        la      $t1, student1                   #why is it student1 in practice and not _student???
        addi    $t1, $t1, PROGRAM_OFFSET        #adding the offset to access program

        #save a value to the program section.
        li      $t2, 1521
        sw      $t2, ($t1)
        #load the value back in now.

         .data
# struct _student{
#        int id;
#        char family[20];
#        char given[20];
#        int program;
#        double wam;
# }

student1: .space 56
```

student_struct.s in tutorial code -> week 4

MIPS functions tut.q3

Some key points:

- All registers you use **except S registers** are “clobbered” or uninitialised upon using JAL
  - this means don’t use t-registers “across” JALs!
- All functions must push and pop $ra
- if function X uses any S registers, X must push and pop every register it uses
- returning a value from a function is done through $v0

Sample usage of functions and S registers

```
1         .text
2
3   main__prologue:
4         # should only include begin and push commands and maybe moving arguments (a0, a1 etc..) into S registers
5         # as neccessary.
6         begin
7         push    $ra
8         push    $s0
9         push    $s2
10        push    $s1
11
12
13  main__body:
14        #your code goes here
15
16        move    $s0, $a0        #inputs to a function come in from a0, a1, a2 etc... in the order it is listed.
17        li      $s1, 0          #for each S register you use, push and pop it.
18        li      $s2, 1
19        add     $t5, $s1, $s2   #don't push and pop t, a, v registers
20
21
22
23        move    $v0, $t5        #output of a function comes out from $v0
24
25
26  main__epilogue:
27        #there should be no code here except pushing and popping. loading return value into V0
28        #can also be acceptable.
29        #pop in the reverse order of push
30        pop     $s1
31        pop     $s2
32        pop     $s0
33        pop     $ra
34        end
35        jr      $ra     #there should be no other code between "end" and "jr $ra"
```

MIPS STYLE!

Tutrial question 6

If have time tut9

Lab question?

Questions and Answers