Advanced Array Manipulation

Written by : cc

1. As in high level languages , when programming in assembly language you should split up your program into smaller functions, that you can reuse. One of the key ideas with functions is that you can call them from any where and return back to where you called the function from. The MIPS processor has two instructions that enable you to call functions, jr and jal.

J -- *Jump*

Description:	Jumps to the calculated address
Operation:	PC = nPC; nPC = (PC & 0xf0000000) | (target << 2);
Syntax:	j target
Encoding:	`0000 10ii iiii iiii iiii iiii iiii iiii`

JAL -- *Jump and link*

Description:	Jumps to the calculated address and stores the return address in $31
Operation:	$31 = PC + 8 (or nPC + 4); PC = nPC; nPC = (PC & 0xf0000000) | (target << 2);
Syntax:	jal target
Encoding:	`0000 11ii iiii iiii iiii iiii iiii iiii`

JR -- *Jump register*

Description:	Jump to the address contained in register $s
Operation:	PC = nPC; nPC = $s;
Syntax:	jr $s
Encoding:	`0000 00ss sss0 0000 0000 0000 0000 1000`

2. Syscall Input/Output

Service	System Call Code	Arguments	Result
print integer	1	$a0 = value	(none)
print float	2	$f12 = float value	(none)
print double	3	$f12 = double value	(none)
print string	4	$a0 = address of string	(none)
read integer	5	(none)	$v0 = value read
read float	6	(none)	$f0 = value read
read double	7	(none)	$f0 = value read
read string	8	$a0 = address where string to be stored $a1 = number of characters to read + 1	(none)
memory allocation	9	$a0 = number of bytes of storage desired	$v0 = address of block
exit (end of program)	10	(none)	(none)
print character	11	$a0 = integer	(none)
read character	12	(none)	char in $v0

Example :
Using Jal Instruction

```
.data
    str : .asciiz "Selamat datang di Lab 3 POK !"
.text
.globl main

main:
    jal message
    li $v0, 10
    syscall
message:
    la $a0, str
    li $v0, 4
    syscall
    jr $ra
```

Exercise :

After learning MIPS Language before, Yolo is assigned a task by his teacher named Momo. The task is quite simple, after Momo teach the class about jump instruction Momo wants Yolo to translate the code below to MIPS Language with jal instruction.

```
array = [1, 100, 88, 10, 89, 19, 15, 17]

print('MIN Value :', min(array)) #dengan hasil output "MIN Value : 1"
print('MAX Value :', max(array)) #dengan hasil output "MAX Value : 100"
print('MEAN Value :', sum(array)//len(array)) #dengan hasil output "MEAN Value : 42"
```

MIN will be stored in register **$s1**.
MAX will be stored in register **$s2**.
MEAN will be stored in register **$s3**.

Specification:

- Input:
  - An array of **positive numbers** (don't forget to include/define the array in your code!)

- Output:
  - min (stored in **$s1**) and string to show the result
  - max (stored in **$s2**) and string to show the result
  - mean(stored in **$s3**) and string to show the result

- Save the file with format: Lab03_[Kode Asdos][Nama][NPM].asm

References

1. http://user.it.uu.se/~justin/Teaching/NewDarkSlides/lecture5.pdf
2. http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html
3. http://students.cs.tamu.edu/tanzir/csce350/reference/syscalls.html