DHA Suffa University
Department of Computer Science
Computer Organization & Assembly Language
Spring 2017
Lab # 05 (Indirect and Based Addressing)

Objective:

To deal with arrays using indirect and based addressing in MIPS.

For reference you can visit:

https://www.cs.uregina.ca/Links/class-info/201/SPIM-AddressingMode/lecture.html

https://www.cise.ufl.edu/class/cda5155fa16/protected/MIPS_addressing.pdf

https://www.youtube.com/watch?v=h-HBipu_1P0

Direct Addressing:

The simplest addressing mode is the register addressing. Instructions using registers execute fast because they do not have the delay associated with memory access. But, the number of registers is limited since only 5-bits are reserved to select a register.

Register addressing is a form of direct addressing. The value in the register is an operand instead of being a memory address to an operand.

For example: **add $t0, $t1, $t2**

Indirect and Based Addressing:

- Used only with load and store instructions

load address:

```
la      $t0, var1
```

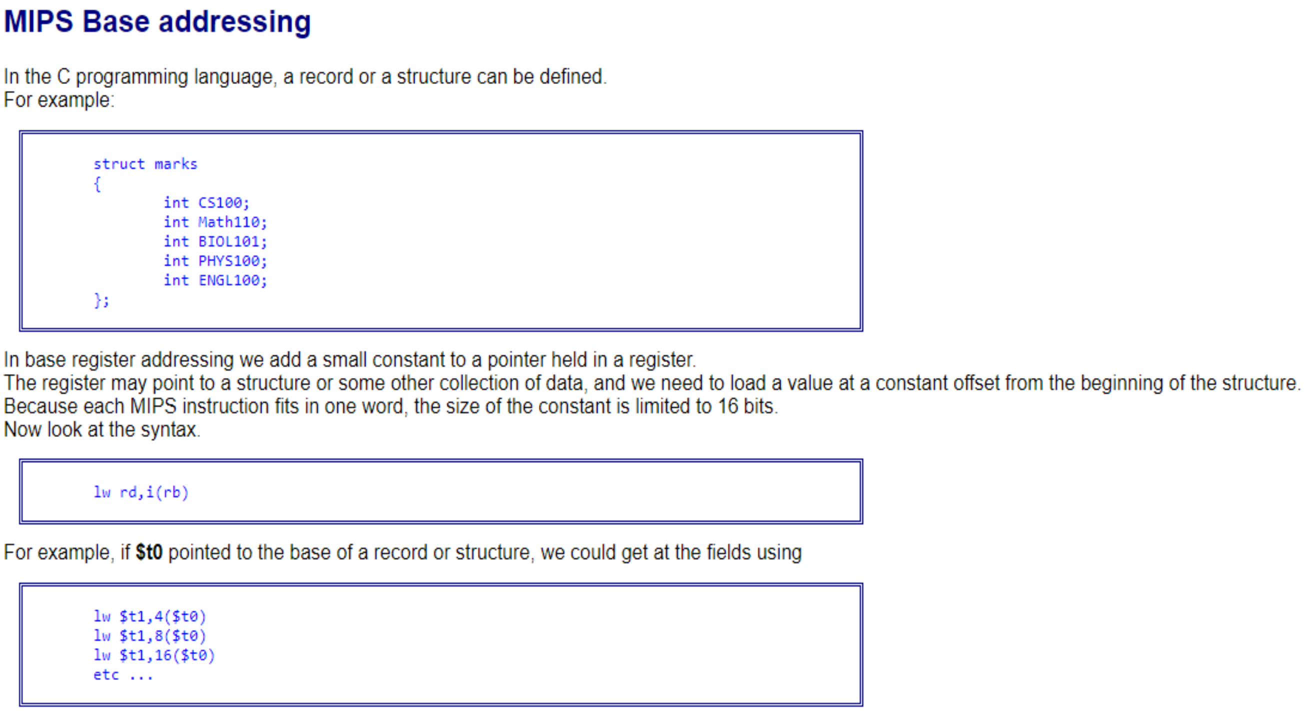
- copy RAM address of var1 (presumably a label defined in the program) into register $t0

Indirect addressing:

lw $t2, ($t0)

- load word at RAM address contained in $t0 into $t2
  sw $t2, ($t0)
- store word in register $t2 into RAM at address contained in $t0

Based or indexed addressing:



MIPS Base addressing

In the C programming language, a record or a structure can be defined.
For example:

```
struct marks
{
        int CS100;
        int Math110;
        int BIOL101;
        int PHYS100;
        int ENGL100;
};
```

In base register addressing we add a small constant to a pointer held in a register.
The register may point to a structure or some other collection of data, and we need to load a value at a constant offset from the beginning of the structure.
Because each MIPS instruction fits in one word, the size of the constant is limited to 16 bits.
Now look at the syntax.

```
lw rd,i(rb)
```

For example, if **$t0** pointed to the base of a record or structure, we could get at the fields using

```
lw $t1,4($t0)
lw $t1,8($t0)
lw $t1,16($t0)
etc ...
```

lw $t2, 4($t0)

- load word at RAM address ($t0+4) into register $t2
- "4" gives offset from address in register $t0
  sw $t2, -12($t0)
- store word in register $t2 into RAM at address ($t0 - 12)
- negative offsets are fine

Note: based addressing is especially useful for:

- arrays; access elements as offset from base address
- stacks; easy to access elements at offset from stack pointer or frame pointer

Example of indirect indexing:

```
.data
num: .word 12,14,16,18,20
.text
.globl main
main:
                li $t1, 0
                li $t2, 5
                la $t0, num     int *a=&b          #  load base address of array into register $t0

Loop:
beq $t1, $t2, exit
lw $t3,($t0)                    a=*b
li $v0, 1
move $a0, $t3
syscall

addi $t1, $t1, 1
addi $t0, $t0, 4
b Loop

exit:
li $v0, 10
syscall
```

Example of based indexing:

```
.data
array1: .space    12                # declare 12 bytes of storage to hold array of 3 integers
.text
.globl main
main:
                  la     $t0, array1              #  load base address of array into register $t0
                  li     $t1, 5                   #  $t1 = 5   ("load immediate")
                  sw     $t1, ($t0)               #  first array element set to 5; indirect addressing
                  li     $t1, 13                  #   $t1 = 13
                  sw     $t1, 4($t0)              #  second array element set to 13
                  li     $t1, -7                  #   $t1 = -7
                  sw  $t1, 8($t0)
la $t4, array1


lw $t1, 0($t4)
li $v0, 1
move $a0, $t1
syscall

lw $t1, 4($t4)
li $v0, 1
move $a0, $t1
syscall

lw $t1, 8($t4)
li $v0, 1
move $a0, $t1
syscall

li $v0, 10
syscall
```

LAB TASK

(1) Write the MIPS code for the following C code:

```
 main()
 {
   int arr[5] = {11,12,13,14,15};
   for (int i=0;i<5;i++)
   {
             arr[i] = arr[i] * 10;
             printf("value of arr[%d] is %d \n", i, arr[i]);
   }
 }
```

(2) Write the MIPS code for the following C code:

```
void main ()
{
  char array[]="DHA Suffa";
  for (int i=9;i>=0;i--)
    printf("%c",array[i]);
}
```