CENG3420

Lab 1-2: Control Instructions and Function Call

Bei Yu

Department of Computer Science and Engineering
The Chinese University of Hong Kong

byu@cse.cuhk.edu.hk

Spring 2018

香港中文大學
The Chinese University of Hong Kong

Last Time

Basic instructions

- `la lw sw li`
- `add addi sub`

Variable Definition

- `.data`
- `.word .asciiz`

Assembly Program Structure

- `.globl .data .text`
- exit `syscall`

Dealing with an Array

Declaration

```
.data
a .word 1 2 3 4 5
```

Remark

- Similar to the definition of array in C++, "`a`" is the address of the first element of the array.
- Rest of the elements can be accessed through `la` with offset.
  (What should be the offset for the 2^{nd} element in the array above?)

Control Instructions I

Jump Registers

Jump to the address contained in register.
Usage: `jr <Register>`

Jump and Link

Jumps to the calculated address and stores the return address in $ra
Usage: `jal <target>`
Operation: Update PC

Jump

Jumps to the calculated address.
Usage: `j <target>`

Control Instructions II

Usage

```
.data
# your variables
.text
main:

#your instructions
jal f  #now jump to branch label f
li $v0, 10
syscall

f: #your instructions
la #...
lw #...
jr $ra  #return to the position next to where jal happens.
```

Control Instructions III

Branch on Equal

Usage: `beq <reg1>, <reg2>, <target>`
Description: If the values stored in `reg1` and `reg1` are equal, jump to `target`.

Branch on Less Than

Usage: `blt <reg1>, <reg2>, <target>`
Description: If the value stored in `reg1` is smaller than that in `reg1`, jump to `target`.

Branch on Greater Than

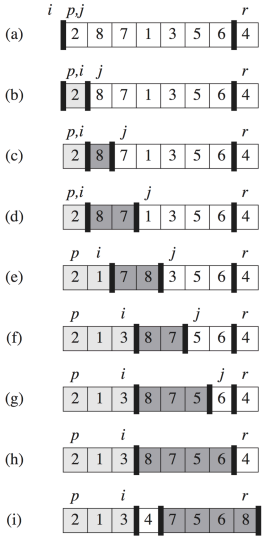
Usage: `bgt <reg1>, <reg2>, <target>`

Partitioning

- Pick an element, called a pivot, from the array.
- Reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way).

```
1:  function Partition(A, lo, hi)
2:      pivot ← A[hi]
3:      i ← lo-1;
4:      for j = lo; j ≤ hi-1; j ← j+1 do
5:          if A[j] ≤ pivot then
6:              i ← i+1;
7:              swap A[i] with A[j];
8:          end if
9:      end for
10:     swap A[i+1] with A[hi];
11:     return i+1;
12: end function
```

Example of Partition()



∗In this example, p = lo and r = hi.

Assignment

An array `array1` contains the sequence `-1 22 8 35 5 4 11 2 1 78`. Rearrange the element order in this array such that,

1. All the elements smaller than the 3^{rd} element (i.e. 8) are on the left of it,
2. All the elements bigger than the 3^{rd} element (i.e. 8) are on the right of it.

Submission Method:

Submit your source code "`<name-sid>.s`" onto blackboard.

Appendix-A simple Sort Example

Swap v[k] and v[k+1]

Assume `$a0` stores the address of the first element and `$a1` stores k.

```
swap: sll $t1, $a1, 2
      add $t1, $a0, $t1
      lw  $t0, 0($t1)
      lw  $t2, 4($t1)
      sw  $t2, 0($t1)
      sw  $t0, 4($t1)
```

Appendix-A simple Sort Example

C style bubble sort:

```
void sort(int v[], int n)
{
    int i,j;
    for(i=0;i<n;i+=1)
    {
        for(j=i-1;j>=0 && v[j]>v[j+1]; j-=1)
        {
            swap(v,j);
        }
    }
}
```

Appendix-A simple Sort Example

Assembly style bubble sort:

Saving registers:

```
sort: addi $sp, $sp, -20
      sw   $ra, 16($sp)
      sw   $s3, 12($sp)
      sw   $s2, 8($sp)
      sw   $s1, 4($sp)
      sw   $s0, 0($sp)
```

Appendix-A simple Sort Example

Assembly style bubble sort:

Move parameters		move	$s2, $a0 # copy parameter $a0 into $s2 (save $a0)
		move	$s3, $a1 # copy parameter $a1 into $s3 (save $a1)
Outer loop		move	$s0, $zero# i = 0
	for1tst:	slt	$t0, $s0,$s3 #reg$t0=0if$s0Š$s3(iŠn)
		beq	$t0, $zero, exit1# go to exit1 if $s0 Š $s3 (i Š n)
Inner loop		addi	$s1, $s0, -1# j = i - 1
	for2tst:	slti	$t0, $s1,0 #reg$t0=1if$s1<0(j<0)
		bne	$t0, $zero, exit2# go to exit2 if $s1 < 0 (j < 0)
		sll	$t1, $s1, 2# reg $t1 = j * 4
		add	$t2, $s2, $t1# reg $t2 = v + (j * 4)
		lw	$t3, 0($t2)# reg $t3 = v[j]
		lw	$t4, 4($t2)# reg $t4 = v[j + 1]
		slt	$t0, $t4, $t3 # reg $t0 = 0 if $t4 Š $t3
		beq	$t0, $zero, exit2# go to exit2 if $t4 Š $t3
Pass parameters and call		move	$a0, $s2 # 1st parameter of swap is v (old $a0)
		move	$a1, $s1 # 2nd parameter of swap is j
		jal	swap # swap code shown in Figure 2.25
Inner loop		addi	$s1, $s1, -1# j -= 1
		j	for2tst # jump to test of inner loop
Outer loop	exit2:	addi	$s0, $s0, 1 # i += 1
		j	for1tst # jump to test of outer loop

Appendix-A simple Sort Example

Assembly style bubble sort:

Exit and restoring registers:

```
exit1:
lw   $ra, 16($sp)
lw   $s3, 12($sp)
lw   $s2, 8($sp)
lw   $s1, 4($sp)
lw   $s0, 0($sp)
addi $sp, $sp, 20
```