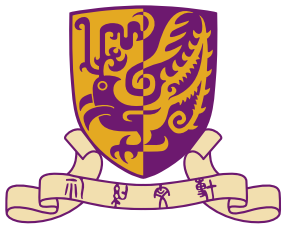
CENG3420
Lab 1-2: Control Instructions and Function Call

Haoyu YANG

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

hyyang@cse.cuhk.edu.hk

2017 Spring



香 港 中 文 大 學
The Chinese University of Hong Kong

Last Time

Basic instructions

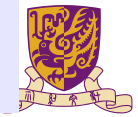
- `la lw sw li`
- `add addi sub`

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

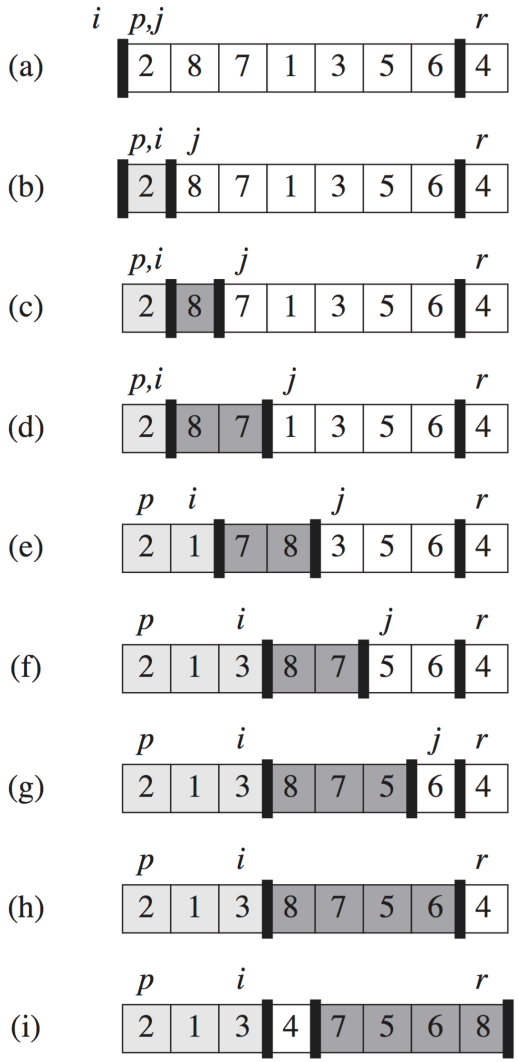
Usage: `bgt <reg1>, <reg2>, <target>`

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.

Lab report should include source code, console result. Add comments to your source code for major steps.