CENG3420
Lab 1-3: Quick Sort

Yang BAI

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

ybai@cse.cuhk.edu.hk

Spring 2021

香港中文大學
The Chinese University of Hong Kong

Last Time

Array Definition

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

`a` is the address of the first element

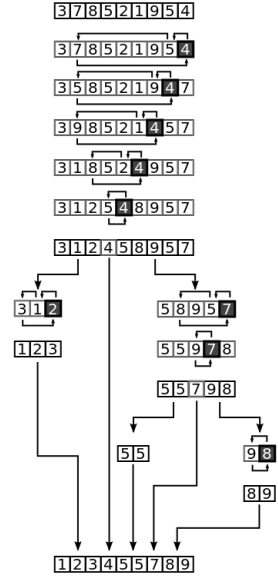
Branch

`jal` (PC is stored in `ra` if you speicify `ra`)
`j jr beq blt bgt`

Quick Sort

Quick Sort Overview

Quicksort is a divide and conquer algorithm. Quicksort first divides a large array into two smaller sub-arrays: the low elements and the high elements. Quicksort can then recursively sort the sub-arrays.

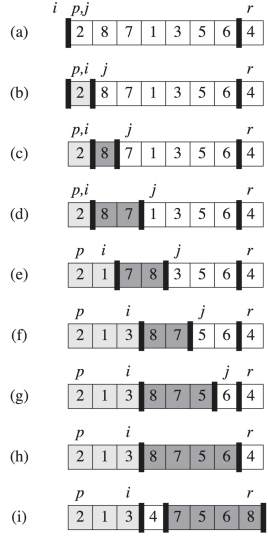


Quick Sort: Partitioning

- 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.

Quick Sort: Sorting

- Recursively apply the partition to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values.

```
1: function QUICKSORT(A, lo, hi)
2:     if lo < hi then
3:         p ← partition(A, lo, hi);
4:         quicksort(A, lo, p - 1);
5:         quicksort(A, p + 1, hi);
6:     end if
7: end function
```

Compiling a Recursive Procedure

A procedure for calculating factorial

```
int fact (int n)
{
    if (n < 1) return 1;
    else return (n * fact (n-1));
}
```

- A recursive procedure (one that calls itself!)

```
fact (0) = 1
fact (1) = 1 * 1 = 1
fact (2) = 2 * 1 * 1 = 2
fact (3) = 3 * 2 * 1 * 1 = 6
fact (4) = 4 * 3 * 2 * 1 * 1 = 24
. . .
```

- Assume n is passed in `a0`; result returned in `ra`

Compiling a Recursive Procedure (cont.)

```
fact:
    addi  sp, sp, -8      # adjust the stack pointer
    sw    ra, 4(sp)       # save the return address
    sw    a0, 0(sp)       # save the argument `n`
    slti  t0, a0, 1       # test for `n` < 1
    beq   t0, zero, L1    # if `n` >=1, go to L1
    addi  t1, zero, 1     # else return 1 in t1
    addi  sp, sp, 8       # adjust stack pointer
    jr    ra              # return to caller
L1:
    addi  a0, a0, -1      # `n` >=1, so decrease n
    jal   fact            # call fact with (n-1)
                          # this is where fact returns
bk_f:
    lw    a0, 0(sp)       # restore argument n
    lw    ra, 4(sp)       # restore return address
    addi  sp, sp, 8       # adjust stack pointer
    mul   t1, a0, t1      # t1 = n * fact(n-1)
    jr    ra              # return to caller
```

Lab Assignment

Quick Sort the following array in ascending order:

Sort the array for this assignment

`-1 22 8 35 5 4 11 2 1 78`

Submission Method:

Prepare a package onto blackboard, including

- All source codes (`<name-sid>-lab1-x.asm`)
- A lab report (`<name-sid>-lab1.pdf`) with step-by-step algorithm of quicksort and all console results.
- Deadline: 23:59, 7 March