CS 2461
Lab- Week 10

1

Today….

- More assembly – using Subroutines
- Homework 4 and Project 4 …write out your flowchart

2

2

Subroutines & Traps in LC3

- TRAP routines
  - System calls to process I/O (or other system tasks)
  - Written by system, called by user
    - Resides as part of system code
  - Steps: Call, Process, Return
- Subroutines – i.e., functions
  - Written by user
  - Called by user program using JSR or JSRR instruction
  - Returns to calling program (callee) – using RET instruction
  - Steps: Call, Process, Return

3

3

Using Subroutines

- In order to use a subroutine, a programmer must know:
  - its address (or a label that will be bound to its address)
  - its function (what does it do?)
    - NOTE: The programmer does not need to know *how* the subroutine works, but what changes are visible in the machine's state after the routine has run.
  - its arguments (where to pass data in, if any) – which registers
  - its return values (where to get computed data, if any)- which register
- User code must save registers used to pass arguments
  - If subroutine uses other registers, then save them before use and restore before returning
- Example: SUB
  - Inputs are in registers R1, R2
  - Output is in R3

4

4

Changing Subtraction code to a Subroutine

- need to be able to call and return from SUB subroutine
- inputs are in R1,R2 and Output is in R3= R2 – R1

give label to first line in the code…this is address of subroutine SUB…To call, the user program needs to set PC to this address

```
SUB    NOT R0, R1  ; complement R1 and add 1 to get
       ADD R0, R0, #1      ; 2's complement R2 = -R1
       ADD R3, R0, R2 ;  R3= R0+R2 = R2 – R1
       RET    ; replace HALT by RET to return to caller
```

5

5

Using SUB from 'main'

- main code: subtract two numbers in memory and write back difference.
  - Read two numbers from memory locations number1, number2 and store into registers R1, R2.
  - Call SUB and store result in memory location result

```
    .ORIG x3000
    LD R0, number1
    LD R1, number2
    JSR SUB ; call SUB (JSRR if SUB not within 2^10)
    ST R3, result; store result returned in R3 into memory
    HALT
Number1   .FILL x000A
Number2   .FILL #8
Result    .BLKW #1     ;reserve space for result
```

If R2 is used in main then need to save them into memory

6

6

```
; what if address of SUB is not within 11 bit offset?
.ORIG x3000
Loop        LD R1, number1 ; load number1 into R1
            LD R2, number2 ; load number2 into R2
            ST R3, SaveR3   ; save register R3
            LD R5, goSUB    ; load address of SUB into R5
            JSRR R5         ; go to subroutine whose address in R5
            STR  R3, result ; store result of SUB…it returned value in R3
            LD R3, SaveR3           ; restore old value R3
            HALT
number1     .FILL #10
number2     .FILL # -8

goSUB       .FILL  SUB ; initialize goSUB to address of SUB

SaveR3      .BLKW 1; reserve space for SaveR3
result      .BLKW #1
SUB         NOT R0, R1
            ADD R0, R0, #1   ; R0 = -R1
            ADD R3, R0, R2
            RET
            .END
```

7

7

Example: subroutine2.asm

- Replace each element A[i] in an array with A[i]-X until A[i] is <=0
  - Call subroutine SUB to compute subtraction A[i] –X
  - Assume X is stored at some memory location
- Open subroutine2.asm in LC3

```
i=0;
while ( A[i] >= 0) {
      A[i] = X - A[i];
      i=i+1; }
```

A[i] = SUB(A[i], X);

8

8

Multiplication using repeated addition

- No Multiplication operation in LC3….implement MULT using repeated addition

```
;assume non-zero positive x,y
while (x>0) {
       mult = mult + y;
       x= x-1;
```

LC3 code outline

Multiply the values in registers R1, R2 and return in R3

```
; initially clear R3 and check if either X or Y is zero
ADD R0, R2, #0  ; copy R2 to R0
loop   BRnz done
       ADD R3, R3, R1 ; add to product
       ADD R0, R0, # -1 ; decrement X
       BRnzp loop
done   HALT   ; program halts with product in R3
```

9

9

Exercise -Multiplication

Write LC3 code to compute product of array element with X

Replace SUB in example with MULT

```
i=0;
while ( A[i] >= 0) {
       A[i] = MULT(X,A[i];
       i=i+1; }
```

Write MULT subroutine
Assume for now that X,Y are positive…
but work on implementing general version later
…need it for HW4/project!

LC3 code outline for multiplication

Multiply the values in registers R1, R2 and return in R3

```
; initially clear R3 and check if either X or Y is zero
ADD R0, R2, #0  ; copy R2 to R0
loop   BRnz done
       ADD R3, R3, R1 ; add to product
       ADD R0, R0, # -1 ; decrement X
       BRnzp loop
done   HALT   ; program halts with product in R3
```

10

10

HW4 and Project 4 … status and questions ?

- have you read HW4 ?
- have you read Project 4 ?

- Design flowchart for Project 4
  - Main interface
  - How do you read the input Key ?
  - How do you read the input message (to be encrypted) ?
  - Which subroutines to call to encrypt ?
  - Which ones to call to decrypt ?

- Get started ASAP…. You cannot complete this project if you start on it two days before it is due!
  - Homework 4 subroutines are used in Project 4

11