LC3 Assembly Programming
Week 8 Lab
Exercise

1

Today

- Write assembly program
  - Loop through memory locations
    - Get familiar with Load and Store instructions
  - Implement XOR operation
- The lab exercises, and assembly programming homework, will help you prepare for Project 4
  - Please save the code you write – you can reuse it for the project
  - XOR operation is required for the project

2

2

Questions on Writing assembly ?

- Label = address of that instruction
- Specify immediate values as decimal (#), hex (x), binary (b)
- .FILL : declare and initialize a value at the mem location
  - Count .FILL #5 reserves memory location with label Count and sets its value to 5…. Analogous to C statement `int Count=5;`
- `.BLKW :` declare a variable at the memory location
  - Sum .BLKW #1 reserves one location with label Sum
    - Analogous to C statement int Sum;
  - Array .BLKW #10 reserves 10 locations with label Array at the starting address of the 10 consecutive locations
- .ORIG
- .END
- HALT

3

3

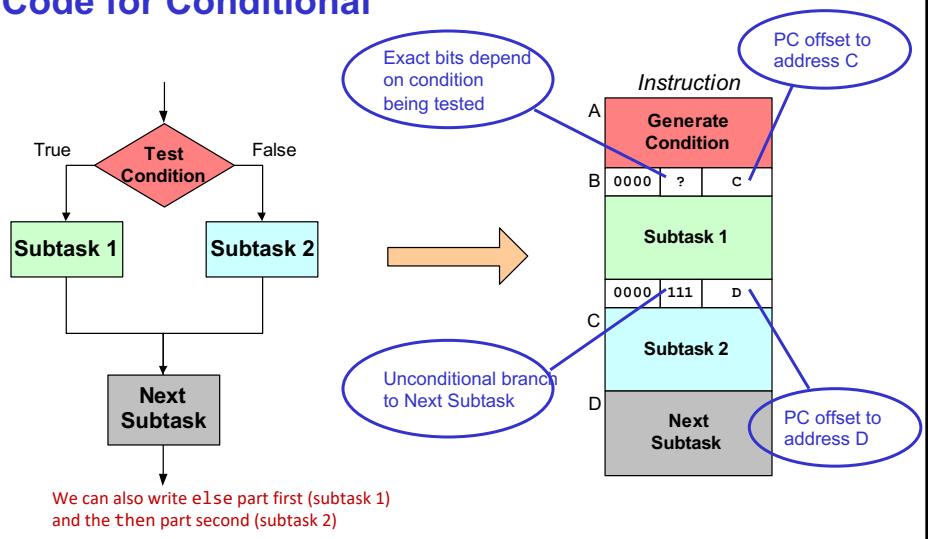
Questions on Writing assembly ?

- Conditional statements – use Branch instruction and condition code registers
  - Condition code registers set by value of data from previous instruction
  - ADD R1, R2, R3 -- output R1 out of ALU determines N, Z, P conditions
- IF statement…..
- While statement (iterate/loop)

4

4

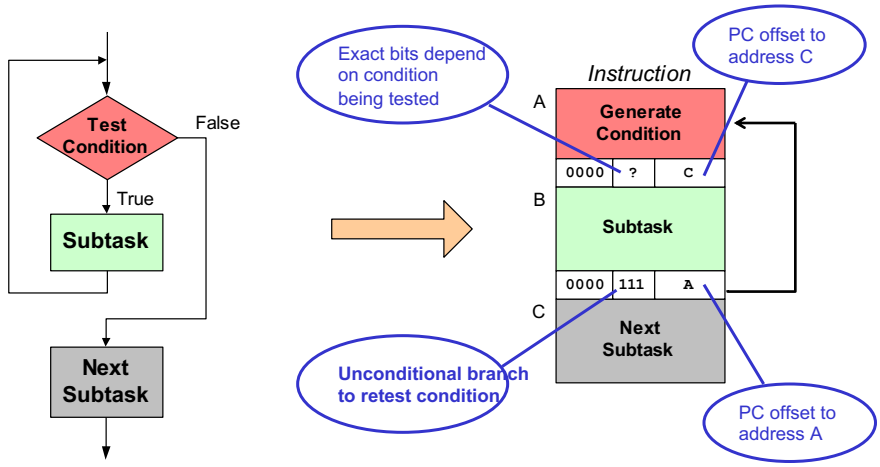
Code for Conditional



Assuming all addresses are close enough that PC-relative branch can be used. 5

5

Code for Iteration



Assuming all addresses are on the same page.

6

Using LC3 Assembler/Simulator (LC3 Tools)

- Important Note on using LC3 simulator
  - Open file (or start new one), type your assembly code
  - Assemble
  - Open Simulator…. **Set breakpoint at last instruction/HALT of your program– this will stop the simulation after the instruction**
  - **Set it by click on small dot (exclamation) next to where you want to set breakpoint…it goes red to indicate breakpoint set.**
- To execute one instruction at a time, use step over
  - Track the register and memory contents after each step
- or run entire program, and check contents at end of the program
- "Step into" option shows you execution within each instruction (will get clearer once we cover datapath details)

7

7

Creating and Loading a "data" file

- LC3Tools permits loading multiple object files
  - Loaded at the address specified in that object file (i.e., .ORIG command)
- Can use this to create and load a file containing the data to be processed by your code.
- Ex: MyArray.asm is a list of numbers starting at address x4000
- Assemble the code – creates object code MyArray.obj
- Load this object file into simulator
  - Important: make sure you reset program counter is set to start of your main program.

MyArray.asm

```
.ORIG       Put values 10,20,30,40
x4000       at addresses x4000, 4001, 4002, 4003
.FILL #10   Respectively.
.FILL #20   Loading MyData.obj will result in these
.FILL #30   Values in those memory addresses of
.FILL #40   the simulator
.END
```

8

8

Creating and Loading a "data" file – alternate option

- LC3Tools permits specifying multiple .ORIG and .END code blocks in same file
- When you assemble, it creates one object file and loads each code block (data) into the specified starting locations.
- Example: a list of numbers starting at address x4000
- You can type in these lines after (or before) your main program…which should also have a .ORIG and a .END command
- When you assemble the code it creates one object file to be loaded into simulator

```
;start of you program
.ORIG x3000
 ; instructions in your program
.END ; signifies end of instructions
```

```
.ORIG x4000
.FILL #10
.FILL #20
.FILL #30
.FILL #40
.END
```

Start your data block here at x4000
Put values 10,20,30,40
at addresses x4000, 4001, 4002, 4003
Assembler will load into those memory addresses ofthe simulator

9

9

Exercise: XOR and looping through an array

- Write LC3 assembly program that iterates through an array `MyArray` of N1 numbers and replaces `MyArray[i]` with `MyArray[i] XOR temp`
  - `MyArray` is stored starting at x4000
  - `temp` is a variable in your program – assume it is x22
    - Recall: a variable is a memory location with a label
    - Can be initialized using the .FILL assembler directive

```
temp    .FILL  x22
```

  - N1 is a variable (length of array) in your program – assume N1=10

```
i=N1;  ;initialize register i to N1 =10
j = temp;  /* read from memory location temp into register j */
while i>0 {
      MyArray[i] = MyArray[i] XOR j;
      i=i-1; }  /* in assembly you have to load starting
                /* address of MyArray into a register…
                /* starting address is not equal to i !
```

10

10

Steps/Considerations

- How to implement bitwise XOR
- How to load data from your array into memory into registers
- How to store the updated data from registers into memory
- How to do this in a loop, using a counter
- You can refer to the example from lecture – it has solution now!

```
i=10;  ;initialize register i to 10
j = temp;  /* read from memory location temp into register j */
while i>0 {
       MyArray[i] = MyArray[i] XOR j;
       i=i-1; }  /* in assembly you have to load starting
                 /* address of MyArray into register…
                 /* starting address is not equal to i !
```

11

11

Exercise

- Program to compute: `MyArray[i]= MyArray[i] XOR temp`
  - MyArray is stored starting at x4000 -- MyArray .FILL x4000
  - temp is a variable in your program – assume it is x22 using
    temp .FILL x22
  - N is a variable (length of array) in your program – assume N=10
    ADD R0, R0, #10 will set value of R0=10 … think of R0 as the loop iterator `i`
- How do you implement R1 XOR R2 in LC3 (bitwise XOR)?
  - You have AND and NOT ??
- How do you implement: `for (i=0; i < N; i++)`
  Equivalently (easier to do): `while i >0`
  note: R0 now stores N=10 at start
- To iterate through MyArray:
  - Fetch temp into R1
  - load starting address of MyArray (x4000) into a register R4 – R4 points to current index
  - load element into R2, and XOR with x55 (value of temp stored in R1): R3=R1 XOR R2
  - Loop N times (in this case N =10)
  - Each time (a) decrement counter R0 by 1 and (b) increment register R4 by 1 so it points to next array element

12

12

Exercise: Implementing XOR

- How do you implement R1 XOR R2 in LC3 (bitwise XOR)?
  - You have AND and NOT ??
- Recall: A XOR B = ((NOT A) AND B) OR (A AND (NOT B))
- Oops no OR instruction in LC3 !! … so use DeMorgan's laws to convert OR to AND and NOT
- C OR D = NOT (NOT (C OR D)) = NOT ((NOT C) AND (NOT D))
- In your LC3 program: the two values are in registers R1 and R2
  - You may need to use other registers for temporary values.

13

13

Exercise: implementing conditional statement (While)

- How do you implement: `for (i=0; i < N; i++)`
  Equivalently (easier to do): `while i>0` (initially i=N)
  note: R0 now stores N=10 at start
- Convert to conditional and unconditional branches
  If `i` is not positive then exit loop
  inside the loop decrement i at each iteration
  last instruction of loop is unconditional branch to instruction to test if i is positive

```
WHILE        Branch NZ endwhile
              …..
              i= i-1;            /* i is stored in R0 */
              Branch NZP to WHILE
endWhile
```

14

14

inside the loop: MyArray [i] = MyArray[i] XOR temp

- Before entering loop, load value of `temp` into `R1`
- Use a register (`R4`) to point to current location in array
  - Initially load `MyArray` (value `x4000`) into `R4` – outside the loop
- To iterate through `MyArray`:
  - Load array element into `R2` *use LDR instruction…get MyArray start address into reg*
  - XOR with `x22` (value of temp stored in R1): `R3=R1 XOR R2`
  - Each time (a) decrement counter R0 by 1 and (b) increment register R4 by 1 so it points to next array element

```
            Load value from temp into R1
            Load address of MyArray (x4000) into R4
WHILE       Branch NZ endwhile
            Load from address in R4 into register R2
            R3 = R1 XOR R2;
            Store R3 into address in R4
            R4 = R4 +1 ; point to next element in array
             i= i-1;           /* i is stored in R0 */
             Branch NZP to WHILE
endWhile
```

15

15