Programming Studio #4

ECE 190

Programming Studio #4

- Topics this week:
  - Systematic Decomposition
  - Memory Addressing Modes
- In Studio Assignment
  - LC-3 Programming Assignment

Announcements

- MP1
  - Due Wednesday, February 17th at 5PM
  - Read handout carefully
  - Cheating Policy is on the website
- Exam 1 – Thurs 2/25 – 7pm-9pm
  - Review Session: TBA
  - Report conflicts to professors NO LATER THAN Feb 17th

Systematic Decomposition

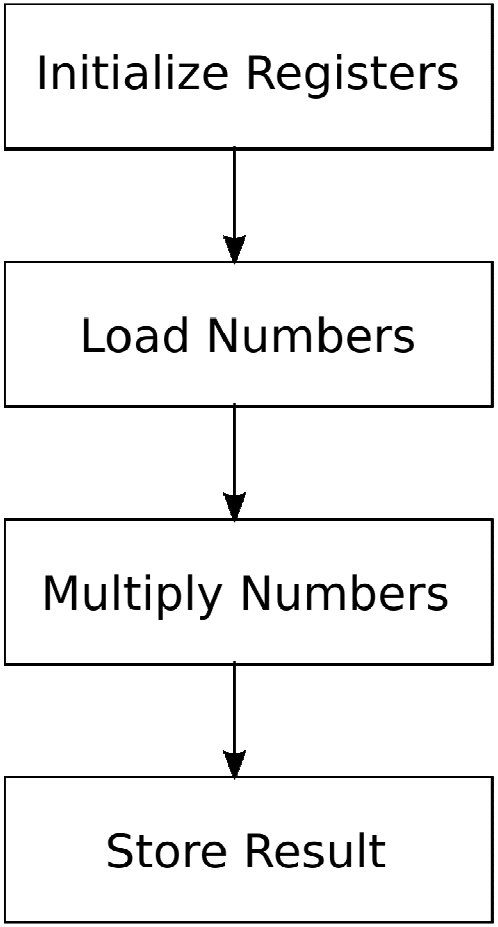
- Do not go straight to coding
- Start with a problem in plain language
  - Break it into smaller steps progressively
  - Continue until steps small enough
  - Finally, steps can be turned into code

Example: Multiplication

- Problem:
  - Design a multiplier program in LC-3 binary. The program should load numbers from x3100, x3101. The numbers are multiplied, and stored back in x3102.
- Given algorithm: Add X to itself Y times.
  - X * Y = Z.
  - 5 x 3 = 5 + 5 + 5
  - Loop using branch instructions.

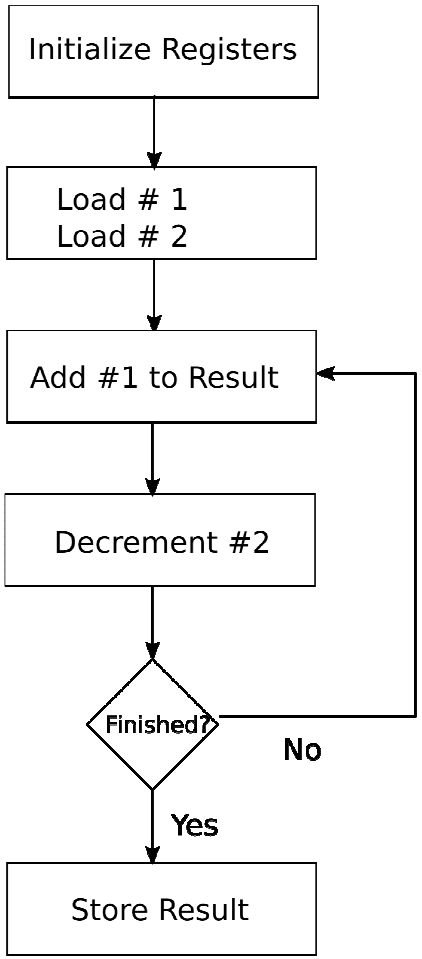
First Phase: Large Steps

- First few steps help show program flow
- For multiplication we have 3 main steps
- Why do we initialize registers?



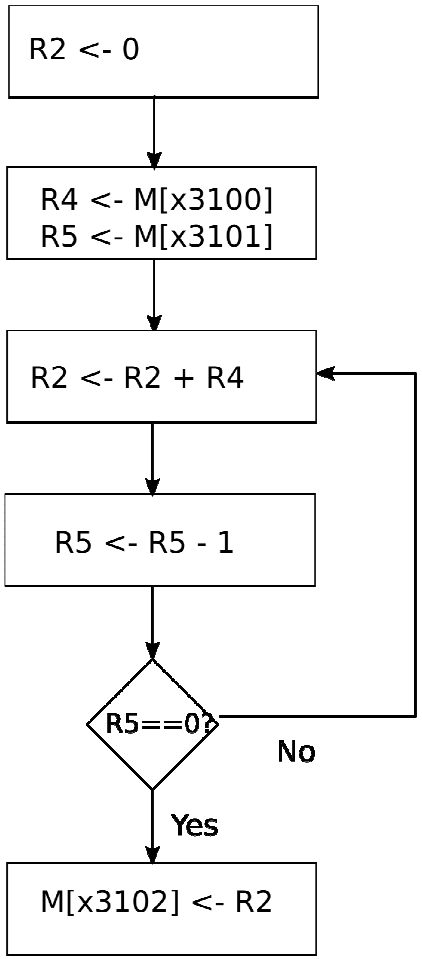
Middle Phase: Small Steps

- Break large steps into smaller steps
- Specify program flow more completely
- Include multiplication algorithm
- Add control flow to allow loop behavior



Final Phase: Pseudo Code

- Take small steps and convert to pseudo
- Choose which registers hold which data
  - R2 stores running sum
  - R4 and R5 to hold numbers
- Move from pseudo instructions into assembly instructions.
- Why don't we need to initialize R4 or R5?



ASM and Machine Code

```
0011 0000 0000 0000
0101 010 010 1 00000  ;x3000 AND R2,R2,#0 ; Initialize
0010 100 0 1111 1110  ;x3001 LD R4, FE ;x3100 load first value
0010 101 0 1111 1110  ;x3002 LD R5, FE ; x3101 load second value
0001 010 010 0 00 100 ;x3003 ADD R2,R2,R4 ;Start of loop, running sum
0001 101 101 1 11111  ;x3004 ADD R5,R5,#-1 ; Decrement the count
0000 001 1 1111 1101  ;x3005 BRp  x3003 ; Check if R5 is zero
0011 010 0 1111 1011  ;x3006 ST R2, F9 ; store result
1111 0000 0010 0101   ;x3007 HALT
```

Addressing Modes

- Need to move data in/out of registers from/to memory
- Several ways to specify the memory location
  - PC Relative
  - Base + Offset
  - Indirect
  - LEA

PC Relative

- Syntax:
  - LD Rx, imm_val | Rx <= MEM[PC[1]+imm_val]
- Uses the incremented PC of the current instruction plus an immediate value to calculate address:
  - Address = PC + 1 + imm_val
- imm_val is sign extended to 16 bits
- Used when data is in a single location close to the instruction
- Example:

Address	Contents
x3005	LD R1, x3
x3006	....
x3007	....
x3008	....
x3009	x22

- R1 receives contents of MEM[x3006+x3] = MEM[x3009] = x22

Base + Offset

- Syntax:
  - LDR Rx, Ry, imm_val | Rx <= MEM[Ry+imm_val]
- Uses the contents of a register plus immediate value to calculate address:
  - Address = Ry + imm_val
- Used when data is too far to access with PC Relative mode, or when accessing many values stored close to an address not known when writing the program
- Example: (Assuming R2 holds the value x5000)

Address	Contents
x3005	LDR R1, R2, x0
x3005	LDR R3, R2, x1
…	….
x5000	x1A
x5001	x22

- R1 receives contents of MEM[x5000+x0] = x1A
- R3 receives contents of MEM[?] = ?
- **NOT USED TO COPY CONTENTS FROM ONE REGISTER TO THE OTHER!!!**

Indirect

- Syntax:
  - LDI Rx, imm_val | Rx <= MEM[MEM[PC[1]+imm_val]]
- Uses the incremented PC of the current instruction plus an immediate value to calculate an address. Then reads the contents of that memory location and uses them as the address from where actual data is accessed:
  - Address = value at location [PC + 1 + imm_val]
- imm_val is sign extended to 16 bits
- Used when data is in a single location far from the instruction
- Example:

Address	Contents
x3005	LDI R1, x1
x3006	....
x3007	x4000
…	….
x4000	x22

- R1 receives contents of MEM[MEM[x3006+x1]] = MEM[MEM[x3007]] = MEM[x4000] = x22

Load Effective Address

- Syntax:
  - LEA Rx, imm_val | Rx <= PC[1]+imm_val
- Uses the incremented PC of the current instruction plus an immediate value to calculate address and places it in the register:
  - Address = PC + 1 + imm_val
- imm_val is sign extended to 16 bits
- Used when we need to load an address into a register (for use in LDR/STR)
- Example:

Address	Contents
x3003	xBE
x3004	xEF
x3005	LEA R1, -3
x3006	LDR R0, R1, x0
x3007	LDR R2, R1, x1

- R1 receives x3005+1-3 = x3003, R0 Receives?, R2 Receives?

Programming Studio Assignment

- Sum a list of N numbers and store result in x4000
  - Load N from address x3100
  - The list of numbers starts at x3101
  - Use systematic decomposition to break the problem into steps, then create the machine code.
  - You may assume that N >0.
    - Hint: Use LD/ST, LDR/STR, LDI/STI, LEA to your advantage