

Computer Science 210 s1c
Computer Systems 1
2008 Semester 1
Lecture Notes

Lecture 16, 10Apr08:
Assembly Language

James Goodman



Credits: Slides prepared by Gregory T. Byrd, North Carolina State University

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

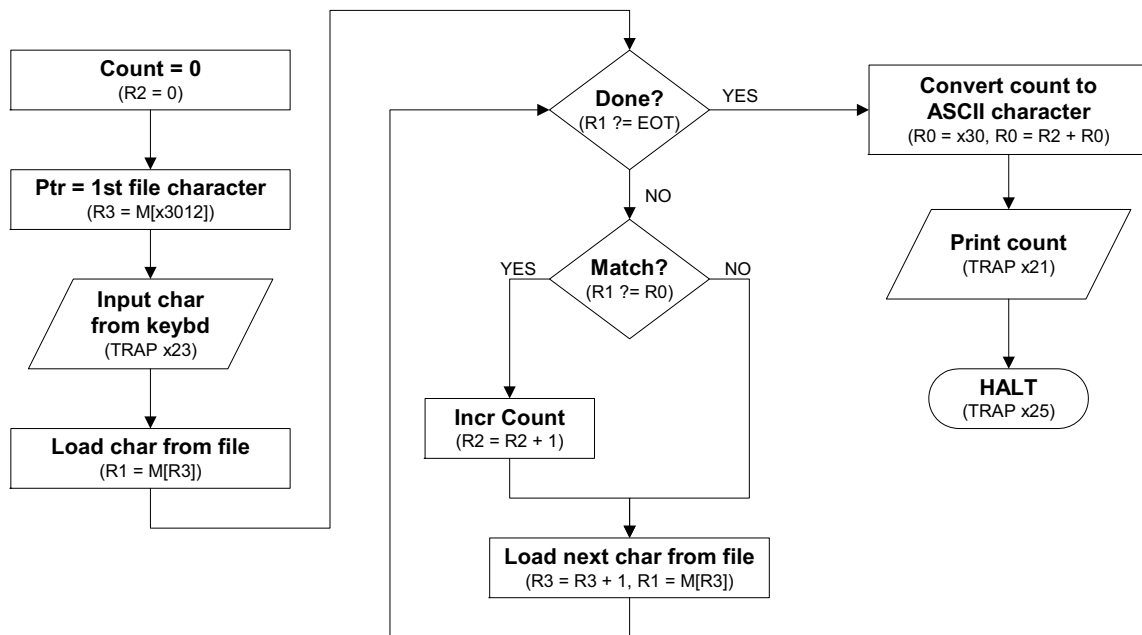
Style Guidelines

Use the following style guidelines to improve the readability and understandability of your programs:

1. Provide a program header, with author's name, date, etc., and purpose of program.
2. Start labels, opcode, operands, and comments in same column for each line. (Unless entire line is a comment.)
3. Use comments to explain what each register does.
4. Give explanatory comment for most instructions.
5. Use meaningful symbolic names.
 - Mixed upper and lower case for readability.
 - ASCIItoBinary, InputRoutine, SaveR1
6. Provide comments between program sections.
7. Each line must fit on the page -- no wraparound or truncations.
 - Long statements split in aesthetically pleasing manner.

Sample Program

Count the occurrences of a character in a file.
Remember this?



10-Apr-08

CS210

275

Char Count in Assembly Language (1 of 3)

```

;
; Program to count occurrences of a character in a file.
; Character to be input from the keyboard.
; Result to be displayed on the monitor.
; Program only works if no more than 9 occurrences are found.
;
;
; Initialization
;
        .ORIG    x3000
        AND     R2, R2, #0          ; R2 is counter, initially 0
        LD      R3, PTR             ; R3 is pointer to characters
        GETC    R0                  ; R0 gets character input
        LDR     R1, R3, #0          ; R1 gets first character
;
; Test character for end of file
;
TEST    ADD     R4, R1, #-4          ; Test for EOT (ASCII x04)
        BRz     OUTPUT              ; If done, prepare the output
  
```

10-Apr-08

CS210

276

Char Count in Assembly Language (2 of 3)

```

;
; Test character for match.  If a match, increment count.
;
        NOT        R1, R1
        ADD        R1, R1, R0 ; If match, R1 = xFFFF
        NOT        R1, R1     ; If match, R1 = x0000
        BRnp       GETCHAR    ; If no match, do not increment
        ADD        R2, R2, #1

;
; Get next character from file.
;
GETCHAR ADD        R3, R3, #1 ; Point to next character.
        LDR        R1, R3, #0 ; R1 gets next char to test
        BRnzp      TEST

;
; Output the count.
;
OUTPUT  LD         R0, ASCII  ; Load the ASCII template
        ADD        R0, R0, R2 ; Covert binary count to ASCII
        OUT        ; ASCII code in R0 is displayed.
        HALT       ; Halt machine

```

10-Apr-08

CS210

277

Char Count in Assembly Language (3 of 3)

```

;
; Storage for pointer and ASCII template
;
ASCII   .FILL      x0030
PTR     .FILL      x4000
        .END

```

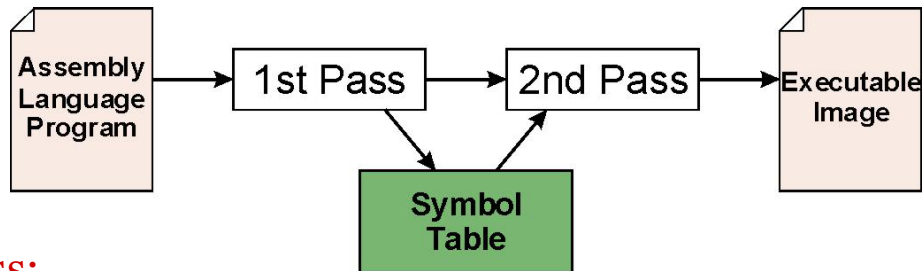
10-Apr-08

CS210

278

Assembly Process

Convert assembly language file (.asm)
into an executable file (.obj) for the LC-3 simulator.



First Pass:

- scan program file
- find all labels and calculate the corresponding addresses;
this is called the *symbol table*

Second Pass:

- convert instructions to machine language,
using information from symbol table

First Pass: Constructing the Symbol Table

1. Find the **.ORIG** statement,
which tells us the address of the first instruction.
 - Initialize location counter (LC), which keeps track of the current instruction.
2. For each non-empty line in the program:
 - a) If line contains a label, add label and LC to symbol table.
 - b) Increment LC.
 - NOTE: If statement is **.BLKW** or **.STRINGZ**,
increment LC by the number of words allocated.
3. Stop when **.END** statement is reached.

NOTE: A line that contains only a comment is considered an empty line.

Practice

Construct the symbol table for the program in Figure 7.1 (Slides 7-11 through 7-13).

Symbol	Address