

# Notes: LC-3 Assembler Language

## A. Why?

- Assembler language is easier to read/write than machine language.

## B. Outcomes

By the end of the class you should

- Know the LC-3 assembler version of the data calculation, movement, and control instructions.
- Know what the .ORIG, .FILL, .BLKW, and .END pseudoinstructions do and how to use them.

## C. Chapter 7: LC-3 Assembler Language

- Machine code ugly; assembler language less ugly.
- Each line of assembly language is translated into a single machine language instruction by an Assembler program.
- An assembler provides
  - Mnemonic labels for opcodes
  - Labels: symbolic names for address locations
  - Automatic conversion of binary / hex / decimal
  - Pseudo-operations to reserve memory, mark beginning/end of program, etc.
- Instruction format:
  - Format (label and ; comments are optional)  
 LABEL      OPCODE    OPERANDS ; COMMENTS
  - Label: Symbolic name for a memory location; can be used as PC+offset operand in load, store, add, and, etc.
  - Opcode: Symbolic name for the 4-bit opcode: ADD, AND, BR, LD, LDR, LDI, LEA, NOT, ST, STR, STI, TRAP go with the instructions we've seen so far.
    - Branch instructions: BRN, BRZ, BRP, BRNZ, BRZP, BRNP, BRNZP, BR followed by location to go to. (BR equivalent to BRNZP)
  - Operands: Separated by commas; can include R0 – R7 (registers), symbolic label, x with hex digits (e.g. x03AF) or # with decimal integer (possibly negative), e.g. #-16, #0, #15. (x and # useful for immediate operands)
    - E.g. ZEROR1    AND    R1, R1, #0 ; R1 <- 0
  - Can have a whole line be a comment by starting it with a semicolon.
- Pseudoinstructions (a.k.a. assembler directives)
  - Don't generate machine code; are used by assembler

- Origin: Specifies where the program goes in memory. (Only 1 per program.)  
`.ORIG xNNNN` (where NNNN is 4 hex digits)  
 goes before first line of program (not including comments)
- END: Specifies last line of assembler program file  
`.END`
  - Note: `.END` and halt instructions are different.
- Fill: Defines one word of memory, initializes it.  
`LABEL .FILL VALUE` (VALUE can be xNNNN, #decimal (possibly < 0))
- Block of words: Defines N words of memory initialized to zeros.  
`LABEL .BLKW N` (N is decimal number of words; label attached to first.)

## A. Sample Assembler Program

Here's a sample program; the hex addresses aren't part of the program; they show where each word of code or data will be stored. The value of X (the number to multiply by Y) has to be filled in before

```

; Program to multiply an integer by the number 6
; (Note: Comments and program have been slightly modified)
;
; .ORIG x3050 ; (Start this program at x3050)
x3050 LD R1, Y ; R1 = k where 0 <= k <= Y
x3051 LD R2, X ; Number x to multiply
x3052 AND R3, R3, #0 ; R3 = X * (Y-k)
;
; The inner loop
;
x3053 LOOP ADD R3, R3, R2 ; R3 = X*(Y-(k-1))
x3054 ADD R1, R1, #-1 ; --k
x3055 BRp LOOP ; loop until k = 0
;
x3056 ST R3, RES ; RES = X*(Y-0) = X*Y
x3057 TRAP x25 ; halt
;
x3058 X .FILL #16 ; (constant 16)
x3059 Y .FILL x0006 ; (constant 6)
x305A RES .BLKW 1 ; Holds X*Y at end
;
.END

```

## A. The Assembly Process

- The assembly process should
  - Translate the AL (Assembly Language) program into ML (Machine Language).
    - Each AL instruction yields one ML instruction word.
  - Interpret pseudo-operations correctly.
  - Depends on pseudo-op
- Labels are a problem
  - A label corresponds to a memory location.
  - If an instruction may reference a label that hasn't been encountered yet, the assembler can't form the instruction word. ("Forward" reference to a label.)
- Two-pass assembler:
  - Make first pass to generate a symbol table
    - Maps labels to memory locations
  - Make second pass to generate the machine language instructions
- To generate the Symbol Table
  - Scan each line of assembler program, keeping track of current address.
  - The `.ORIG` tells you what address to start with.
  - Increment address as instructions and pseudo-operations are encountered
    - Enter defined labels into the symbol table.
    - When you see a label, add it to the symbol table along with the current address
  - Stop when you see `.END`