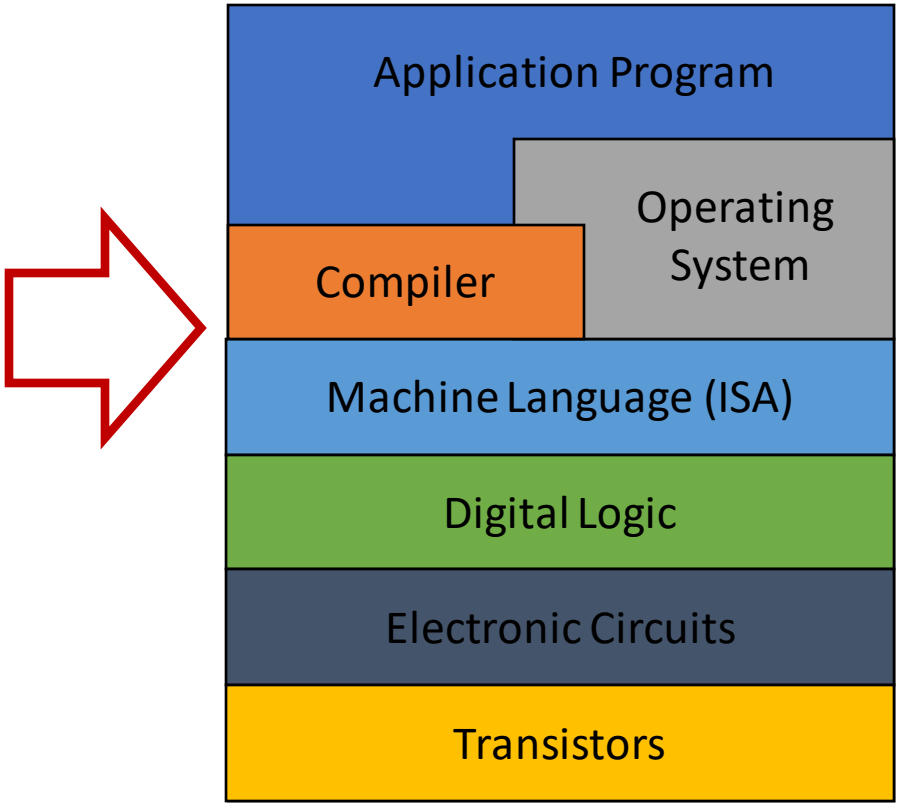
Assembly Language

EECS388 Fall 2022

Context: Moving Up a Level

- Recommended reading

Chapter 7 of “Introduction to Computing,” Patt, Patel



** Picture in the slides are in part taken from “Introduction to Computing From bits & gates to C and beyond” Patt, Patel

Programming Languages

- High level
  - Java, C++, Python, C
  - ISA independent
- Low level
  - Assembly language
  - ISA dependent
  - Use mnemonics for opcode
    - 0001 -> ADD
    - We don’t need to remember instruction format
  - Use symbolic names for memory locations
    - E.g., “LABEL” instead of 16 bit memory address

An LC-3 Assembly Program

Lines start with ";" are comments

```
01  ;
02  ; Program to multiply an integer by the constant 6.
03  ; Before execution, an integer must be stored in NUMBER.
04  ;
05          .ORIG   x3050
06          LD      R1,SIX
07          LD      R2,NUMBER
08          AND     R3,R3,#0
09
0A  ; The inner loop
0B  ;
0C  AGAIN   ADD     R3,R3,R2
0D          ADD     R1,R1,#-1
0E          BRp     AGAIN
0F  ;
10          HALT
11  ;
12  NUMBER  .BLKW   1
13  SIX     .FILL   x0006
14  ;
15          .END
```

Instructions

```
; Clear R3. It will
; contain the product.

; R1 keeps track of
; the iterations
```

Pseudo-ops

Note: the line numbers are not part of the program!

Instructions

Optional

Label Opcode Operands ; Comment

```
AGAIN   ADD   R1,R2,R3
        LD    R4,ADDRESS
        AND   R5,R5,#0   ;clear
```

"#" Decimal "x" Hex "b" binary

Labels

- Symbols used to identify memory locations
- In LC-3: 1 to 20 alphanumeric characters
  - Starting with a letter
  - Except character sequences that cause ambiguity
    - Invalid: ADD, NOT, x1000, R5, 2LA
    - Valid: R2D, LIST, CAMP, C3

Labels (Cont.)

- Reasons for referring to a memory location
  - Target of a branch
  - Location that contains a value in memory

```
05          .ORIG   x3050
06          LD      R1,SIX
07          LD      R2,NUMBER
08          AND     R3,R3,#0        ; Clear R3. It will
09                                  ; contain the product.
0A  ; The inner loop
0B  ;
0C  AGAIN   ADD     R3,R3,R2
0D          ADD     R1,R1,#-1       ; R1 keeps track of
0E          BRp     AGAIN           ; the iterations
0F  ;
10          HALT
11  ;
12  NUMBER  .BLKW   1
13  SIX     .FILL   x0006
```

Pseudo Ops (Assembler Directives)

- Help the assembler translate assembly to machine ISA
  - .ORIG
    - Where to place LC-3 program in memory
  - .FILL
  - .BLKW
  - .STRINGZ

.ORIG

- Where to place LC-3 program in memory

Memory Address

```
0x3050  ;
0x3051  ; Program to multiply an integer by the constant 6.
0x3052  ; Before execution, an integer must be stored in NUMBER.
        ;
        .ORIG   x3050
        LD      R1,SIX
        LD      R2,NUMBER
        AND     R3,R3,#0        ; Clear R3. It will
                                ; contain the product.
        ; The inner loop
        ;
0x3053  AGAIN   ADD     R3,R3,R2
0x3054          ADD     R1,R1,#-1       ; R1 keeps track of
0x3055          BRp     AGAIN           ; the iterations
        ;
0x3056          HALT
        ;
0x3057  NUMBER  .BLKW   1
0x3058  SIX     .FILL   x0006
        ;
        .END
```

.FILL

- Initialize a memory location to a number or label

Memory Address

```
;
; Program to multiply an integer by the constant 6.
; Before execution, an integer must be stored in NUMBER.
;
        .ORIG   x3050
        LD      R1,SIX
        LD      R2,NUMBER
        AND     R3,R3,#0        ; Clear R3. It will
                                ; contain the product.
; The inner loop
;
AGAIN   ADD     R3,R3,R2
        ADD     R1,R1,#-1       ; R1 keeps track of
        BRp     AGAIN           ; the iterations
;
        HALT
;
NUMBER  .BLKW   1
SIX     .FILL   x0006
;
        .END
```

0x3050
0x3051
0x3052
0x3053
0x3054
0x3055
0x3056
0x3057
0x3058

.BLKW

- Set aside a block of memory

Memory Address

```
0x3050  ;
0x3051  ; Program to multiply an integer by the constant 6.
0x3052  ; Before execution, an integer must be stored in NUMBER.
        ;
        .ORIG   x3050
        LD      R1,SIX
        LD      R2,NUMBER
        AND     R3,R3,#0        ; Clear R3. It will
                                ; contain the product.
        ; The inner loop
        ;
0x3053  AGAIN   ADD     R3,R3,R2
0x3054          ADD     R1,R1,#-1       ; R1 keeps track of
0x3055          BRp     AGAIN           ; the iterations
        ;
0x3056          HALT
        ;
0x3057  NUMBER  .BLKW   1
0x3058  SIX     .FILL   x0006
        ;
        .END
```

Number of blocks

.STRINGZ

- Initialize n+1 memory locations with zero extended ASCII code of character in the input strin

```
        .ORIG     x3010
HELLO   .STRINGZ  "Hello, World!"
```

Memory snapshot:

x3010:	x0048	H
x3011:	x0065	e
x3012:	x006C	l
x3013:	x006C	
x3014:	x006F	
x3015:	x002C	
x3016:	x0020	
x3017:	x0057	
x3018:	x006F	
x3019:	x0072	
x301A:	x006C	
x301B:	x0064	
x301C:	x0021	
x301D:	x0000	

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
32	20	40	[space]	64	40	100	@	96	60	140	`
33	21	41	!	65	41	101	A	97	61	141	a
34	22	42	"	66	42	102	B	98	62	142	b
35	23	43	#	67	43	103	C	99	63	143	c
36	24	44	$	68	44	104	D	100	64	144	d
37	25	45	%	69	45	105	E	101	65	145	e
38	26	46	&	70	46	106	F	102	66	146	f
39	27	47	'	71	47	107	G	103	67	147	g
40	28	50	(	72	48	110	H	104	68	150	h
41	29	51	)	73	49	111	I	105	69	151	i
42	2A	52	*	74	4A	112	J	106	6A	152	j
43	2B	53	+	75	4B	113	K	107	6B	153	k
44	2C	54	,	76	4C	114	L	108	6C	154	l
45	2D	55	-	77	4D	115	M	109	6D	155	m
46	2E	56	.	78	4E	116	N	110	6E	156	n
47	2F	57	/	79	4F	117	O	111	6F	157	o
48	30	60	0	80	50	120	P	112	70	160	p
49	31	61	1	81	51	121	Q	113	71	161	q
50	32	62	2	82	52	122	R	114	72	162	r
51	33	63	3	83	53	123	S	115	73	163	s
52	34	64	4	84	54	124	T	116	74	164	t
53	35	65	5	85	55	125	U	117	75	165	u
54	36	66	6	86	56	126	V	118	76	166	v
55	37	67	7	87	57	127	W	119	77	167	w
56	38	70	8	88	58	130	X	120	78	170	x
57	39	71	9	89	59	131	Y	121	79	171	y
58	3A	72	:	90	5A	132	Z	122	7A	172	z
59	3B	73	;	91	5B	133	[	123	7B	173	{
60	3C	74	<	92	5C	134	\	124	7C	174	|
61	3D	75	=	93	5D	135	]	125	7D	175	}
62	3E	76	>	94	5E	136	^	126	7E	176	~
63	3F	77	?	95	5F	137	_	127	7F	177	

.END

- Tells assembler it has reached the end of program
- Just a delimiter marking the end of program for “Assembler” not processor
  - This is different from HALT!

- Looking at the Example Again

```
;
; Program to multiply an integer by the constant 6.
; Before execution, an integer must be stored in NUMBER.
;
        .ORIG   x3050
        LD      R1,SIX
        LD      R2,NUMBER
        AND     R3,R3,#0        ; Clear R3. It will
                                ; contain the product.
; The inner loop
;
AGAIN   ADD     R3,R3,R2
        ADD     R1,R1,#-1       ; R1 keeps track of
        BRp     AGAIN           ; the iterations
;
        HALT
;
NUMBER  .BLKW   1
SIX     .FILL   x0006
;
        .END
```