Computer Science 210 tutorial 5

LC-3 and Assembly code (2)

Tutorial 4 revision

- After tutorial 4, you have learnt
  - How to install LC-3 simulator
  - Edit LC-3 assembly codes in editor
  - Run simple programs
  - Debug LC-3 by using **Step Over** button
- This tutorial will cover:
  - Basic LC3 instructions
  - Inputs and Outputs
  - Branching for IF-ELSE, FOR loop, WHILE loop
  - Subroutines

ADD

Similar to AND

Assembler Formats

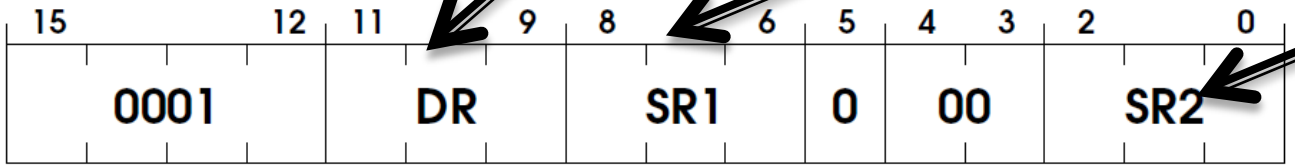
```
ADD   DR, SR1, SR2
ADD   DR, SR1, imm5
```

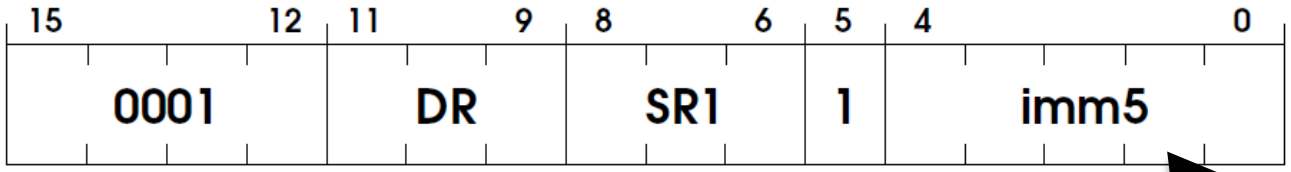
R5 === 101

R2 === 010

R3 === 011

Encodings





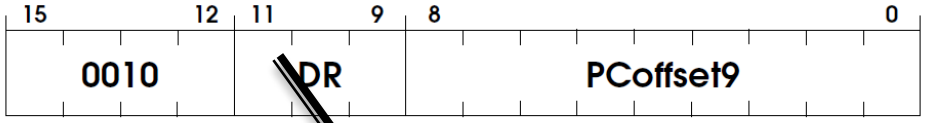
From -16 to +15 === 01111

LD

Assembler Format

LD DR, LABEL

Encoding

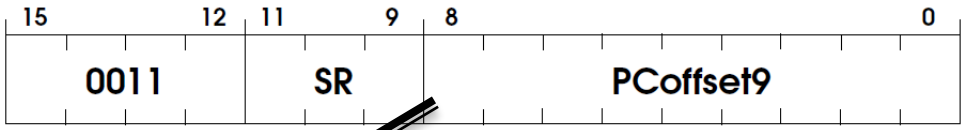


ST

Assembler Format

ST SR, LABEL

Encoding



LC3Edit - 01_ADD_LD.asm

File Edit Translate Help

```
.ORIG x3000

LD      R1,     NUM1            ; R1 <- 10
LD      R2,     NUM2            ; R2 <- 15

ADD     R3,     R1,     R2      ; R3 <- 25 (R1 + R2)

ST      R3,     NUM3            ; 25 -> NUM3

HALT

NUM1    .FILL #10               ; which is 10, can also declare as NUM1 .FILL   #10
NUM2    .FILL #15               ; which is 15, can also declare as NUM1 .FILL   #15
NUM3    .BLKW #1                ; allocate a space

.END
```

Others: LDR, STR, LDI, STI

BR — Conditional Branch

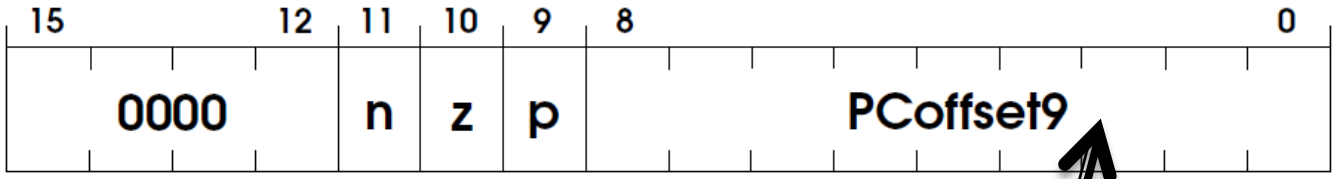
Assembler Formats

Always in nzp order

BRn	LABEL	BRzp	LABEL
BRz	LABEL	BRnp	LABEL
BRp	LABEL	BRnz	LABEL
BR†	LABEL	BRnzp	LABEL

To Implement IF ELSE LOOP

Encoding



How far from current program counter to the LABEL address
So if LABEL is next line === 000000000

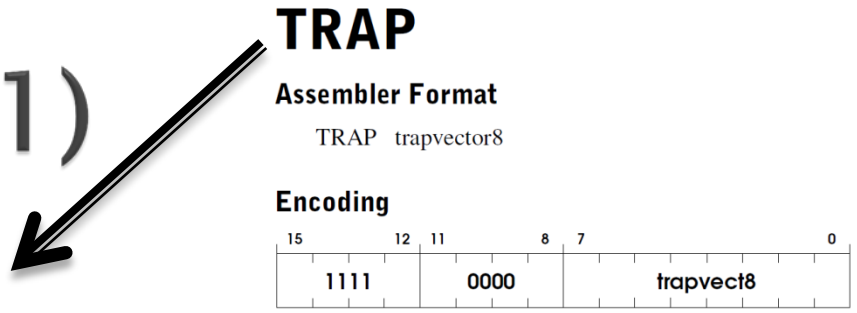
Branch operation

- By getC and Out, you can input 1 character and output 1 character at a time. In order to input and output more, you need loops.
- Loops can be created by using Br (branch operation)
- BR {n|z|p} Label
- BRn branch to Label if register is negative
- BRz branch to Label if register is zero
- BRp branch to Label if register is positive
- → BRzp, BRzn, BRpn…
- BRnzp branch without any condition
- Clearer explanation: http://www.lc3help.com/tutorials/Basic_LC-3_Instructions

ASCII standard to encode characters

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Input/output(1)



- Input and output:
  - Get characters from keyboard to memory/register
  - Print characters from memory/register to screen
- Try running GetC.asm
  - Program does: get 1 input from keyboard and print that out to screen.
- Operations for input/output can be used:
  - Getc
  - Out
  - In
  - Puts

Input/output(2)

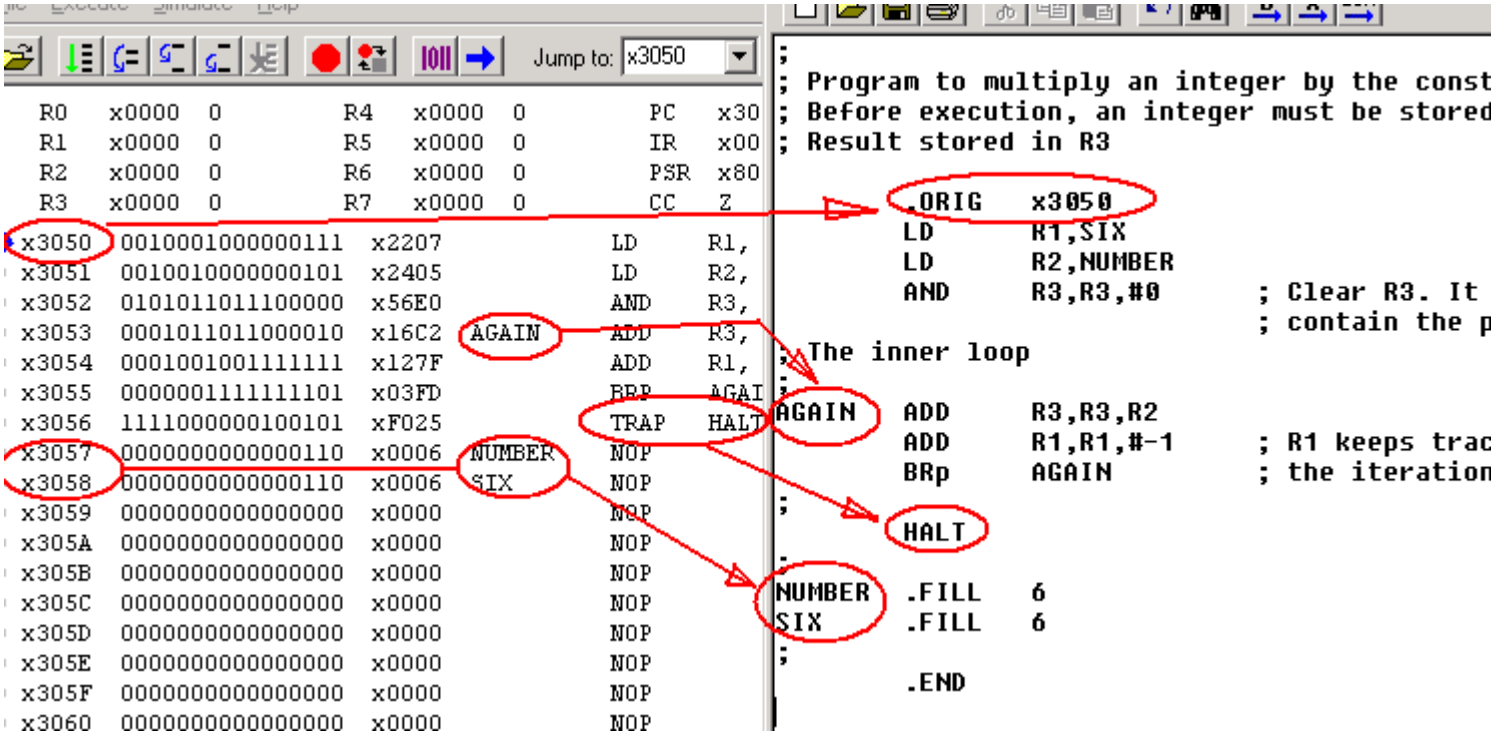
- GetC
  - It takes a character from keyboard
  - Store it in Register R0 (ascii value)
- Out
  - It takes ascii value stored in R0
  - Print the correspondent character out to screen
- In
  - It prints out a line ask user to input
  - It takes a character from keyboard
  - Store it in Register R0 (ascii value)
- Puts
  - It prints out a String
  - Look at printString.asm

Trap Vector	Assembler Name
x20	GETC
x21	OUT
x22	PUTS
x23	IN
x24	PUTSP
x25	HALT

Quick Demo: Chapter 7.1 example (1)

- chapter7_code: 7.1.asm
- What s the program doing?
  - Program multiplies an integer by the constant 6.
  - Before execution, an integer must be stored in NUMBER.
  - Result stored in R3
- Operations used:
  - Ld $(register), VariableName ;load value to register from memory
  - And $(register), $(register), #(decimalNumber) ;bitwise operation
  - BRp Label ;branch (goto) to a Label in memory if register is positive

7.1 example (2)


Jump to: x3050
R0 x0000 0 R4 x0000 0 PC x30
R1 x0000 0 R5 x0000 0 IR x00
R2 x0000 0 R6 x0000 0 PSR x80
R3 x0000 0 R7 x0000 0 CC Z
x3050 0010001000000111 x2207 LD R1,
x3051 0010010000000101 x2405 LD R2,
x3052 0101011011100000 x56E0 AND R3,
x3053 0001011011000010 x16C2 AGAIN ADD R3,
x3054 0001001001111111 x127F ADD R1,
x3055 0000001111111101 x03FD BRP AGAI
x3056 1111000000100101 xF025 TRAP HALT
x3057 0000000000000110 x0006 NUMBER NOP
x3058 0000000000000110 x0006 SIX NOP
x3059 0000000000000000 x0000 NOP
x305A 0000000000000000 x0000 NOP
x305B 0000000000000000 x0000 NOP
x305C 0000000000000000 x0000 NOP
x305D 0000000000000000 x0000 NOP
x305E 0000000000000000 x0000 NOP
x305F 0000000000000000 x0000 NOP
x3060 0000000000000000 x0000 NOP
; Program to multiply an integer by the const
; Before execution, an integer must be stored
; Result stored in R3
.ORIG x3050
LD R1,SIX
LD R2,NUMBER
AND R3,R3,#0 ; Clear R3. It
; contain the p
; The inner loop
AGAIN ADD R3,R3,R2
ADD R1,R1,#-1 ; R1 keeps trac
BRp AGAIN ; the iteration
HALT
NUMBER .FILL 6
SIX .FILL 6
.END

Examples and Exercises

- Create an example to echo an user input, i.e:
  - Hi, what is your name?
  - David Beckham
  - Hi David Beckham, nice to meet you.
- Do exercises:
  - Input a number from 0 to 9
  - Print out all the number from 0 to that number
  - Example:
    - Input: 4
    - Output: 0 1 2 3 4