MACHINE ORGANIZATION & ASSEMBLY LANGUAGE PROGRAMMING

Lab session 023 and 024

Shirin Haji Amin Shirazi

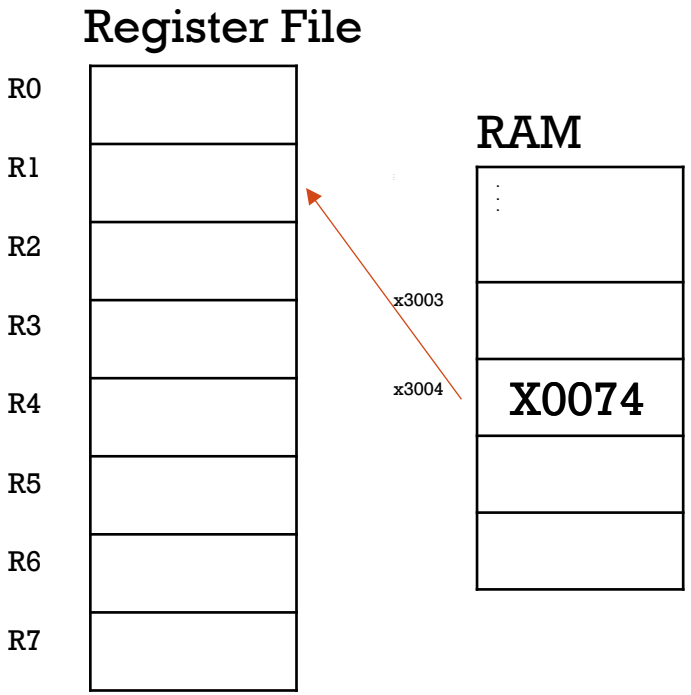
Winter 2021

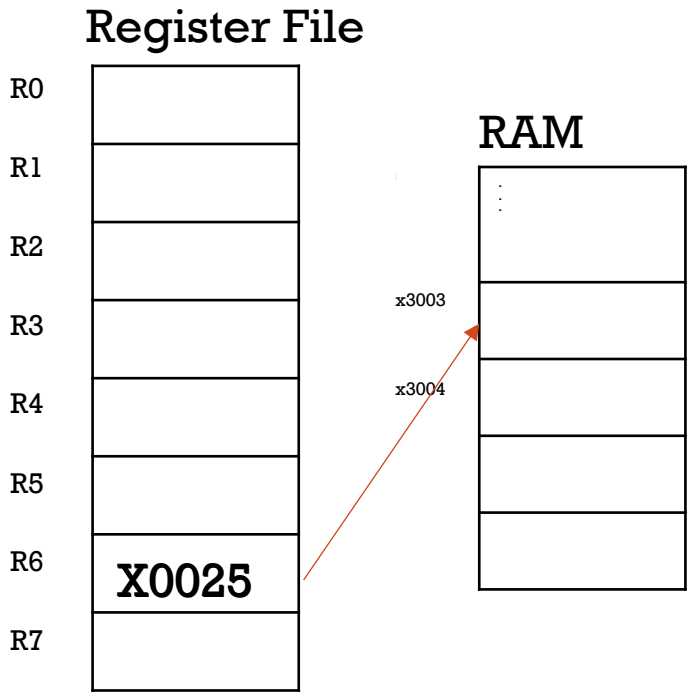
- Arrays
- Loops
- LDR
- Console output

LAB 4

LOAD AND STORE

- Where do we have data?
  - Register file in the cpu
    - R0-R7
  - System Memory (RAM)
- Move data between the two and manipulate them !
- LOAD
- STORE





LOAD / STORE

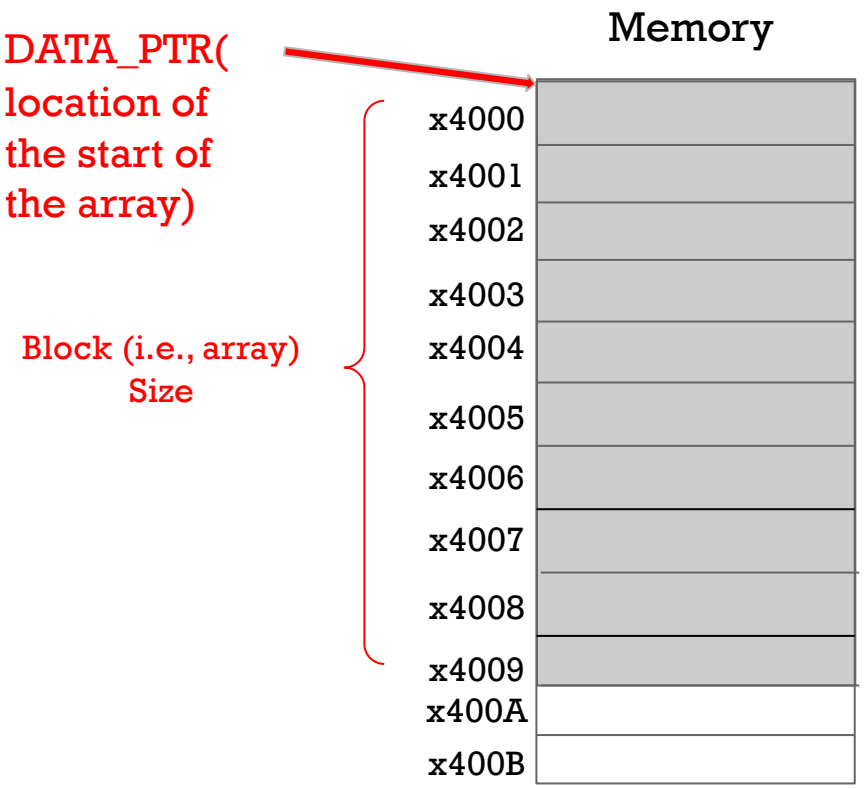
recap

- Load:
  - LD -> direct
    - LD R1, dec_65
  - LDI -> indirect
    - Address in the label
    - LDI R1, dec_65_PTR
  - LDR -> Relative
    - Address in a register
    - Base/offset
    - LDR R1, R3, #4

- Store:
  - ST-> direct
    - ST R1, x4000
  - STI -> indirect
    - Address in the label
    - STI R1, dec_65_PTR
  - STR -> Relative
    - Address in a register
    - Base/offset
    - STR R1, R3, #4

CREATE AN "ARRAY"

- Now we can create an empty "array" to fill in later with:
  - Base address (DATA_PTR)
  - Size (ARRAY_SIZE)



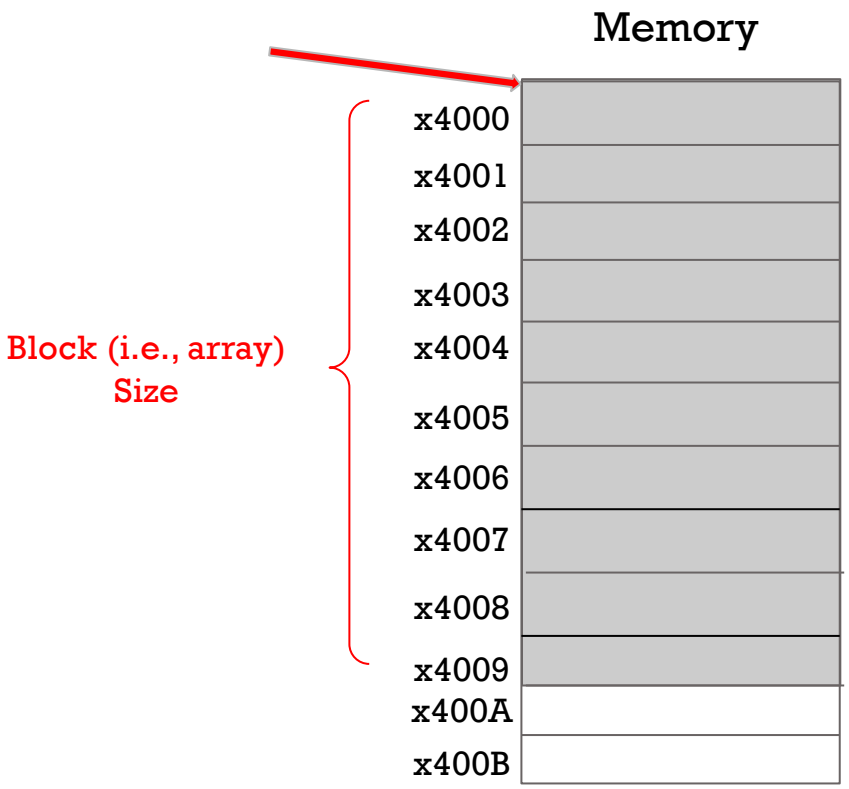

```
DATA_PTR .fill x4000     ; location of array block (i.e., x4000)
ARRAY_SIZE .fill #10     ; size of the array (i.e., block size)
```

ARRAYS

```
;-------------
; Instructions
;-------------
LD R1, Array_PTR
LD R2, SIZE

...

HALT

;-------------
; Local Data
;-------------
Array_PTR .FILL x4000
SIZE .FILL #10

;-------------
; Remote Data
;-------------
.ORIG x4000
.BLKW #10
```

ITERATE THROUGH "ARRAY" (LDR AND STR)

- Now we can create an empty "array" to fill in later with:
  - Base address (DATA_PTR)
  - Length (ARRAY_SIZE)




```
LD R1, DATA_PTR          ; load address to array in register R1
```

```
STR R0, R1, #0       ; store user input into array
```

```
LDR R0, R1, #0       ; load from array
```

```
DATA_PTR .fill x4000     ; location of array block (i.e., x4000)
ARRAY_SIZE .fill #10     ; size of the array (i.e., block size)
```

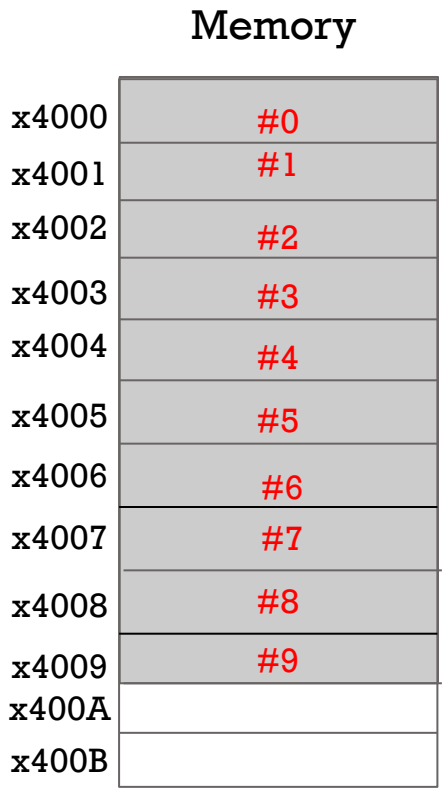
HARD-CODE VS. SOFT CODE

- Soft-Coding :obtaining value/function from external resource
  - Command line
  - Calculations
  - Etc

- Hard-Coding: refers to coding values/functions in source code
  - .FILL
  - etc

SOFT-CODED ARRAY:D

- So, if we hard-code a value (**#0**) into the first address, and get the other values in the rest of the array:
  - How do we print them out?



ASCII ←→ DECIMAL CONVERSION

- **ASCII → Decimal**
  - Subtract x30 or #48
- **Decimal → ASCII**
  - Add x30 or #48

```
LD R3, hex_30    ; R3 <- x30 (add to decimal value to get ASCII)

NOT R4, R3       ; taking 2's complement of x30
ADD R4, R4, #1   ; R4 <- -x30 (add to ASCII to get decimal)
```

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


```
;-------------------------------------------------------------------
;Data
;-------------------------------------------------------------------
hex_30 .fill x30  ; to convert from ASCII to Decimal and back (or #48)
```

POWERS OF 2

$2^n \longrightarrow 2^{n+1}$

$2^n \longrightarrow 2^{n+1} = 2^n 2^1$

$2^{n+1} = 2^n * 2$

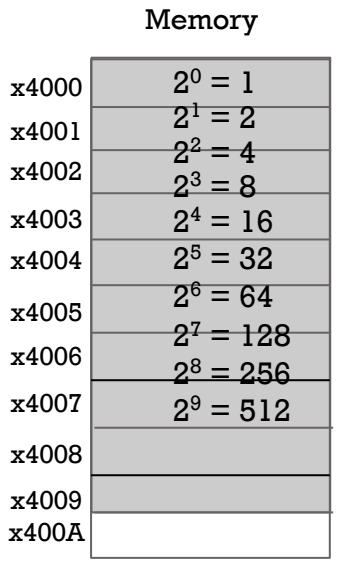
multiply by 2 each time

$2^0 = 1$
$2^1 = 2 = 2$
$2^2 = 2*2 = 4$
$2^3 = 2*2*2 = 8$
$2^4 = 2*2*2*2 = 16$
$2^5 = 2*2*2*2*2 = 32$
$2^6 = 2*2*2*2*2*2 = 64$
$2^7 = 2*2*2*2*2*2*2 = 128$
$2^8 = 2*2*2*2*2*2*2*2 = 256$
$2^9 = 2*2*2*2*2*2*2*2*2 = 512$
...

Multiply

- No Multiplication Instruction in LC3
  - What should we do?
  - LAB2
    - 12*6=12+12+12+12+12+12
    - Addition!
    - **X*2=X+X**
    - **Add the number to itself!**
    - **Hard-code $2^0=1$**
    - **Soft-code the rest**

- Does the previous x30 trick work??
  - Let see!
  - Further assignments and labs
  - In this lab!
    - Print using OUT
    - What do you see?
    - Why???



PRINT

How do we print the multiple digit numbers???

Slide credits to Jason Zellmer

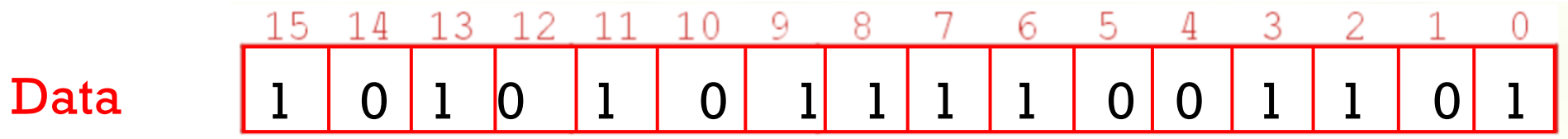
EXERCISES

- Ex1
  - Build an array
  - Hardcode number 0
  - Softcode the rest
  - Store numbers not characters
  - Grab the 7th value,
  - Store it in R2
- Ex2
  - Copy exercise 1
  - Add an output loop
  - Traverse the array
  - Print the characters corresponding to those numbers
- Ex3
  - Similar to ex1
  - Store ten powers of two
  - Grab 7th value, store it in r2
- Ex4
  - Try to print out the values in the array using out!!! Wait what??

ASSIGNMENT3

- Eight registers: R0 - R7
  - All **16-bits wide**
- Instructions
  - All **16-bits wide**
- Data
  - All **16-bits wide**

PRINT THE BINARY REPRESENTATION OF A NUMBER IN MEMORY



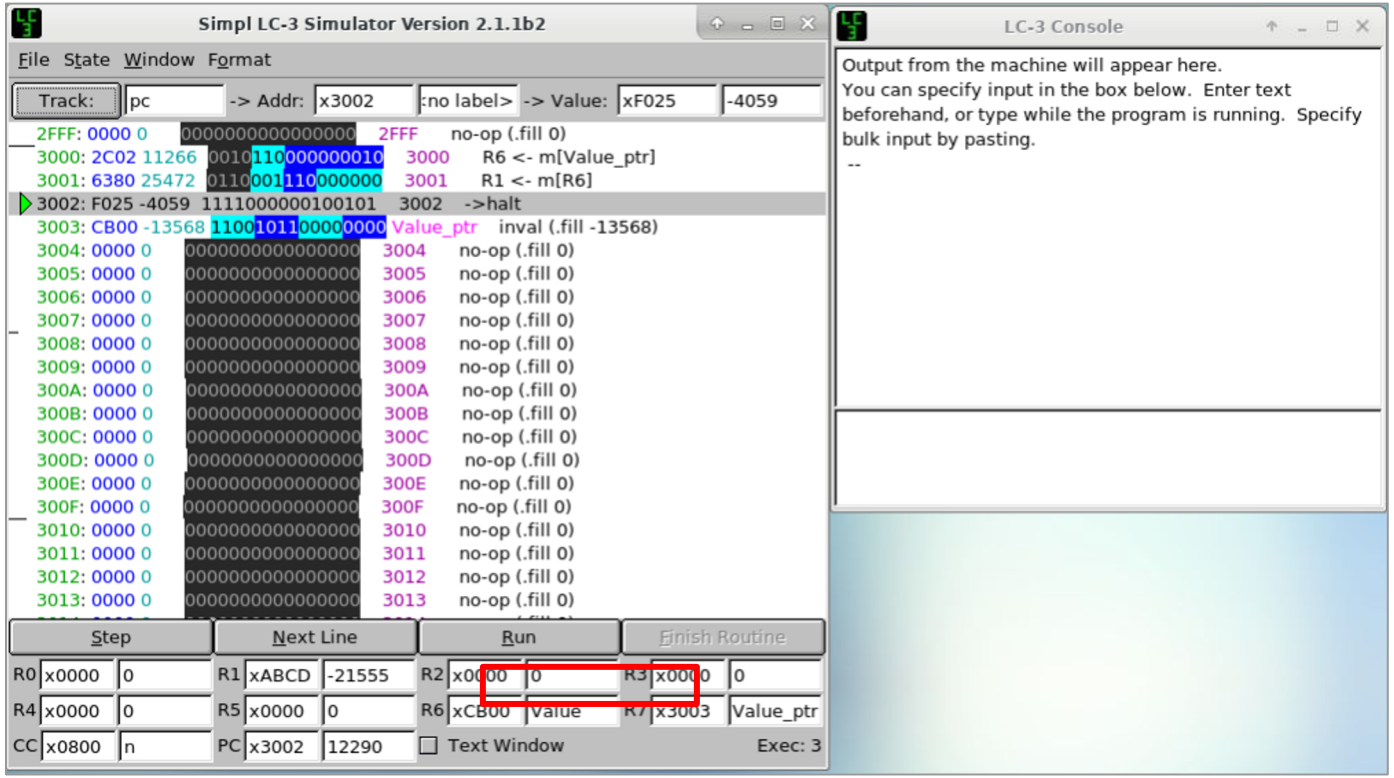
Slide credits to Jason Zellmer

PRINTING BINARY

Data

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	0	1	1	1	1	0	0	1	1	0	1

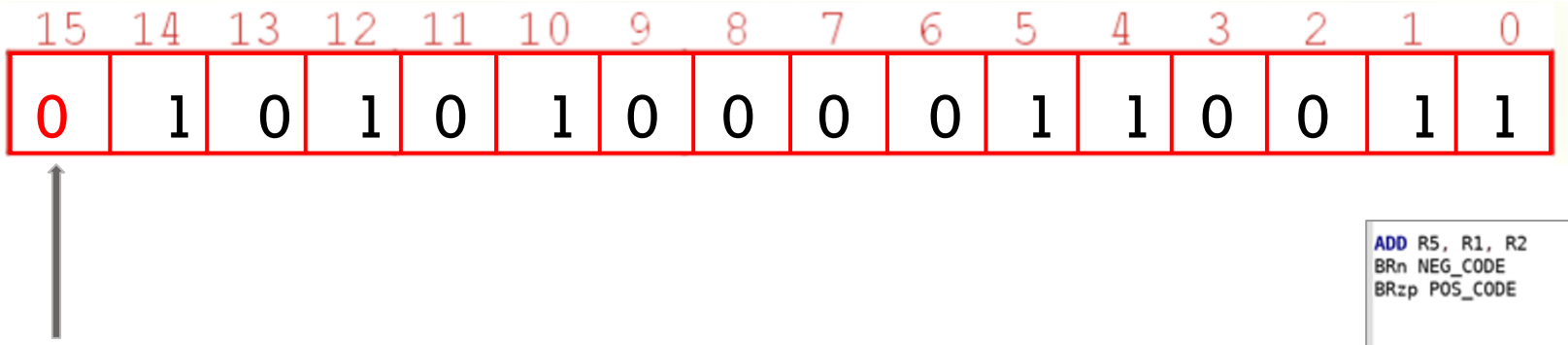
1010 1011 1100 1101 = xABCD



Slide credits to Jason Zellmer

IDEA: PRINT MSB

- So, what if we start by printing out the MSB?
- But how do we do that?



MSB

If MSB=1

the number is negative

Else if MSB=0

the number is positive

```
ADD R5, R1, R2          ; R5 <- R1 - R2
BRn NEG_CODE            ; if result is negative go to neg code
BRzp POS_CODE           ; if result is zero/positive go to pos code


NEG_CODE
    ;
    ; (print '1')
    ;
END_NEG_CODE


POS_CODE
    ;
    ; (print '0')
    ;
END_NEG_CODE
```

HOW TO WE MOVE TO THE NEXT BIT?

- LEFT shift
  - $011101 \xrightarrow{left\ shift} 111010$ (Toss out the MSB, bring 0 in the LSB)
  - $101101 \xrightarrow{left\ shift} 011010$ (Toss out the MSB, bring 0 in the LSB)
  - $00101 \xrightarrow{left\ shift} 01010$ ($5 \xrightarrow{left\ shift} 10$)
    - Left shift == Multiplied by two!
    - Multiplication -> addition
    - Multiply by two == add number to itself
    - ➜ left shift== adding the number to itself