

## Lab 2 – Nikita Berezyuk

February 16, 2022 2:30 PM

# Part 1

Intent: This program adds b and c, stores it in %r0. Then adds a to the %r0 (equal to zero). But, it subtracts b with the previous sum, ultimately making %r2 equal to b - a. Thus, stores %r2 value in variable y and returns.

Qs 1:

! Program 1

! Hint: op3 for subcc is 010100

.begin

! Starts assembling

.org 2048	!Changes location counter to address 2048
dstart .equ 4000	!Tells assembler to equate 4000 to dstart

ld [a], %r1	!Loads data from variable a to register 1	11 00001 000000 00000 1 0111110100000
ld [b], %r2	!Loads data from variable b to register 2	11 00010 000000 00000 1 0111110100100
ld [c], %r3	!Loads data from variable c to register 3	11 00011 000000 00000 1 0111110101000
addcc %r2, %r3, %r0	!Adds binary value data of register 2 and 3, storing the result in register 0	10 00000 010000 00010 0 0000000000011
addcc %r1, %r0, %r4	!Adds binary value data of register 1 and 0, storing the result in register 4	10 00100 010000 00001 0 0000000000000
subcc %r2, %r4, %r2	!Adds binary value data of register 2 and 4, storing the result in register 2	10 00010 010100 00010 0 0000000000100
st %r0, [a]	!Stores (writes) data from register 0 into a.	11 00000 000100 00000 1 0111110100000
st %r2, [y]	!Stores (writes) data from register 2 into y.	11 00010 000100 00000 1 0111110101100
jmp! %r15 + 4, %r0	!Return	10 00000 111000 01111 1 0000000000100

.org dstart	!Changes location counter to address of dstart
a: 15	!Declare a as 15 (0000000001111)
b: 54	!Declare b as 54 (0000000110110)
c: -29	!Declare c as -29 (1111111100011)
y: 0	!Declare y as 0 (0000000000000)

! Stops assembling

.end

Qs 3:

%r4 = 0000000001111)2 or 15)10

- Operation: %r4 = %r1 + %r0 = %r1

PC = 00000188 n z v c  
ireq ET PIL = 0 TBR = ffffffff3  
r0 = 00000000 r1 = 0000000f r2 = 00000027 r3 = ffffffff3  
r4 = 0000000f r5 = 00000000 r6 = 00000000 r7 = 00000000  
r8 = 00000000 r9 = 00000000 r10 = 00000000 r11 = 00000000  
r12 = 00000000 r13 = 00000000 r14 = 00000000 r15 = 00000000  
r16 = 00000000 r17 = 00000000 r18 = 00000000 r19 = 00000000  
r20 = 00000000 r21 = 00000000 r22 = 00000000 r23 = 00000000  
r24 = 00000000 r25 = 00000000 r26 = 00000000 r27 = 00000000  
r28 = 00000000 r29 = 00000000 r30 = 00000000 r31 = 00000000  
Exit Print Load Reload Edit Step Run Stop  
Clear RegFile Clear BreakPts Clear Memory  
Hex Dec Update Screen  
Loc BreakPt MachWord Source Code  
[ 00000184 ] [ 00000000 ] None Prev. 8  
[ 00000188 ] [ 00000000 ] None Prev. 4  
[ 0000018c ] [ 00000000 ] None Prev. 1  
[ 00000190 ] [ 00000000 ] None Next 1  
[ 00000194 ] [ 00000000 ] None Next 4  
[ 00000198 ] [ 00000000 ] None Next 8  
[ 0000019c ] [ 00000000 ] None  
[ 000001a0 ] [ 00000000 ] None  
Loc Offset 00 Offset 04 Offset 08 Offset 0c Prev. 16  
[ 00000000 ] 00000000 00000000 00000000 00000000 Prev. 8  
[ 00000010 ] 00000000 00000000 00000000 00000000 Prev. 4  
[ 00000020 ] 00000000 00000000 00000000 00000000 Next 4  
[ 00000030 ] 00000000 00000000 00000000 00000000 Next 8  
Next 16

# Part 2

Intent: Subtracts the value of x and y. If difference is negative, does a logical or operation of the binary values. If difference is positive, does a logical and operation of the binary values. Then, stores this result in variable a.

Qs 1:

! Program 2

.begin

!Starts assembling

.org 2048	!Changes location counter to start at address 2048
-----------	--

m_start: ld [x], %r1	!m_start label, Loads memory address of variable x into register 1	11 00001 000000 00000 1 01000
ld [y], %r2	!Loads memory address of variable y into register 2	11 00010 000000 00000 1 01000
subcc %r1, %r2, %r0	!Subtracts data from register 1 with register 2 into register 0 (discard)	10 00000 010100 00001 0 00000
bneg else	!If subtraction results in negative value (N flag), jumps to else label	00 0 0110 010 0000000000000000
andcc %r1, %r2, %r3	!If subtraction is positive, does an logic and operation with register 1 and register 2 into register 3	10 00011 010001 00001 0 00000
ba done	!Branch always, will always jump to the done label	00 0 1000 010 0000000000000000
else: orcc %r1, %r2, %r3	!else label, When subtraction is negative, performs logic or operation with register 1 and register 2 into register 3	10 00011 010010 00001 0 00000
done: st %r3, [a]	!done label, Always branches to this, stores data from register 3 into variable a	11 00001 000100 00000 1 01000
impl %r15+4, %r0	!Return	10 00000 111000 01111 1 00000

x: 30	!Declare x as 30 (0000000011110)
y: 48	!Declare y as 48 (0000000110000)
a: 0	!Declare a as 0 (0000000000000)

!Stops assembling

.end

Qs 3:

%r3 = 000000111110)2 or 62)10

- Operation: %r3 = %r1 or %r2

## Part 3

Intent: This program runs through a loop that divides value x in half each time for 4 iterations. It divides value x by shifting it to the right by 1 bit and adding 1 to %r1 each time. After the 4th iteration, stores %r3 in the y variable and returns.

Qs 1:

! Program 3

.begin

!Starts assembling

.org 2048	!Changes location counter to start at address 2048
-----------	--

main: ld [x], %r2	!Main label, Loads memory address of variable x into register 2	11 00010 000000 00000 1 0101110111000
top: subcc %r1, 4, %r0	!Top label, Subtracts register 1 with constant 4 into register 0	10 00000 010100 00001 1 0000000000100
be done	!Branch to label done if equal to zero	00 0 0001 010 0000000000000000000100
srl %r2, 1, %r2	!Shifts register 2 to the right by 1 bit and stores into register 2	10 00010 100110 00010 1 00000000000001
addcc %r1, 1, %r1	!Adds content of register 1 with constant 1 storing it in register 1	10 00001 010000 00001 1 00000000000001
ba top	!Branches always back to label top (loop) until be condition is met	00 0 1000 010 1111111111111111111100
done: st %r3, [y]	!Done label, branches once equal to zero, stores register 3 into memory location y	11 00011 000100 00000 1 0101110111100
jmp1 %r15+4, %r0	!Return	10 00000 111000 01111 1 00000000000100

.org 3000	!Changes location counter to start at address 3000
x: 800	!Declare x as 800 (0001100100000)
y: 0	!Declare y as 0 (0000000000000)

!Stops assembling  
.end

Qs 3:

%r1 = 0000000000100)2 or 4)10

- Operation: %r1 = %r1 + 1 \* 4 times

PC = 00000220 ☐ n ☒ z ☐ v ☐ c

ireq ☐ ET ☐ PIL = 0 TBR = ff000000

r0 = 00000000 r1 = 00000004 r2 = 00000320 r3 = 00000000

r4 = 00000000 r5 = 00000000 r6 = 00000000 r7 = 00000000

r8 = 00000000 r9 = 00000000 r10 = 00000000 r11 = 00000000

r12 = 00000000 r13 = 00000000 r14 = 00000000 r15 = 00000000

r16 = 00000000 r17 = 00000000 r18 = 00000000 r19 = 00000000

r20 = 00000000 r21 = 00000000 r22 = 00000000 r23 = 00000000

r24 = 00000000 r25 = 00000000 r26 = 00000000 r27 = 00000000

r28 = 00000000 r29 = 00000000 r30 = 00000000 r31 = 00000000

Exit Print Load Reload Edit Step Run Stop

Clear RegFile Clear BreakPts Clear Memory

☒ Hex ☐ Dec ☒ Update Screen

Loc	BreakPt	MachWord	Source Code
[ 00000204 ]	<input type="checkbox"/>	00000000	None
[ 00000208 ]	<input type="checkbox"/>	00000000	None
[ 0000020c ]	<input type="checkbox"/>	00000000	None
[ 00000210 ]	<input type="checkbox"/>	00000000	None
[ 00000214 ]	<input type="checkbox"/>	00000000	None
[ 00000218 ]	<input type="checkbox"/>	00000000	None
[ 0000021c ]	<input type="checkbox"/>	00000000	None
[ 00000220 ]	<input type="checkbox"/>	00000000	None

Loc	Offset 00	Offset 04	Offset 08	Offset 0c	
[ 00000000 ]	00000000	00000000	00000000	00000000	Prev. 16
[ 00000010 ]	00000000	00000000	00000000	00000000	Prev. 8
[ 00000020 ]	00000000	00000000	00000000	00000000	Prev. 4
[ 00000030 ]	00000000	00000000	00000000	00000000	Next 4
[ 00000040 ]	00000000	00000000	00000000	00000000	Next 8
[ 00000050 ]	00000000	00000000	00000000	00000000	Next 16

## Part 4

Intent: The purpose of this program is to loop manipulate register 1 three times. %r2 and %r3 do not change, but are used to either perform a logic and/or operation. When %r5 - %r6 is negative, performs "and" operation and adds 1 to %r5 counter. If its not negative, it will perform "or" operation and add 1 to %r6 counter. Will then add 1 top %r4 each time and jump back to top to loop. Once %r4 is 3 / top function three times, program will save %r3 in y and return / break the loop.

- %r4 keeps count of bottom label iterations, %r5 keeps count of else label iterations (and logic), and %r6 tracks amount of times the top program loops (or logic).

Qs 1:

! Program 4

!Starts assembling  
.begin

.org 2048	!Changes location counter to start at address 2048
-----------	--

main: ld [a], %r1	!Main label, loads memory address of variable a into register 1	11 00001 000000 00000 1 0101110111000
ld [b], %r2	!Loads memory address of variable b into register 2	11 00010 000000 00000 1 0101110111100
ld [c], %r3	!Loads memory address of variable c into register 3	11 00010 000000 00000 1 0101110000000
top: subcc %r4, 3, %r0	!Top label, subtracts content from register 4 with constant 3, storing it in register 0 (discard)	10 00000 010100 00100 1 0000000000011

be done	!When equal to zero, branches to done label	00 0 0001 010 000000000000000001010
subcc %r5, %r6, %r0	!If not, subtracts content from register 5 and register 6 into register 0	10 00000 010100 00101 0 0000000000110
bneg else	!If subtraction raises negative flag, branches into else label	00 0 0110 010 00000000000000000000100
orcc %r1, %r2, %r1	!If not, logic or operation between register 1 and 2, storing it into register 1	10 00001 010010 00001 0 00000000000010
addcc %r6, 1, %r6	!Then, adds register 6 with constant 1 into register 6	10 00111 010000 00111 1 00000000000001
ba bottom	!Branches always to bottom label	00 0 1000 010 00000000000000000000011
else: andcc %r1, %r3, %r1	!Else label, performs logic and operation between register 1 and 3 into register 1	10 00001 010001 00001 0 00000000000011
addcc %r5, 1, %r5	!Also, adds content of register 5 with constant 1, storing it into register 5	10 00101 010000 00101 1 00000000000001
bottom: addcc %r4, 1, %r4	!Bottom label, performs addition of content in register 4 with constant 1 into register 4	10 00100 010000 00100 1 00000000000001
ba top	!Will always branch to top label	00 0 1000 010 11111111111111111111010
done: st %r3, [y]	!Done label, stores content from register 3 into variable y	11 00011 000100 00000 1 0101111000100
jmp l %r15+4, %r0	!Return	10 00000 111000 01111 1 00000000000100

.org 3000	!Changes location counter to start at address 3000
a: 0xa0	!Declare a as 0xa0 (0000010100000 or 160)
b: 0x33	!Declare b as 0x33 (0000000110011 or 51)
c: 0x52	!Declare c as 0x52 (0000001010010 or 82)
y: 2	!Declare y as 2 (0000000000010)

!Stops assembling  
.end

Qs 3:  
%r4 = 0000000000011}2 or 3}10

- Operation: %r4 = %r4 + 1 \* 3 times

PC = 00000140

☐ n
☒ z
☐ v
☐ c

ireq

ET

FIL = 0

TBR = ff000000

r0 = 00000000

r1 = 00000033

r2 = 00000033

r3 = 00000052

r4 = 00000003

r5 = 00000001

r6 = 00000002

r7 = 00000000

r8 = 00000000

r9 = 00000000

r10 = 00000000

r11 = 00000000

r12 = 00000000

r13 = 00000000

r14 = 00000000

r15 = 00000000

r16 = 00000000

r17 = 00000000

r18 = 00000000

r19 = 00000000

r20 = 00000000

r21 = 00000000

r22 = 00000000

r23 = 00000000

r24 = 00000000

r25 = 00000000

r26 = 00000000

r27 = 00000000

r28 = 00000000

r29 = 00000000

r30 = 00000000

r31 = 00000000

Exit

Print

Load

Reload

Edit

Step

Run

Stop

Clear RegFile

Clear BreakPts

Clear Memory

Hex

Dec

Update Screen

Loc

BreakPt

MachWord

Source Code

[ 00000124 ]

[ 00000128 ]

[ 0000012c ]

[ 00000130 ]

[ 00000134 ]

[ 00000138 ]

[ 0000013c ]

[ 00000140 ]

Prev. 8

Prev. 4

Prev. 1

Next 1

Next 4

Next 8

Loc

Offset 00

Offset 04

Offset 08

Offset 0c

[ 00000000 ]

[ 00000010 ]

[ 00000020 ]

[ 00000030 ]

Prev. 16

Prev. 8

Prev. 4

Next 4

Next 8

Next 16