! Program 1 ! Hint: op3 for subcc is 010100

! Numbers here are represented in base 10 (decimal)

**.begin** // start of the assembling process

**.org 2048** // tells the assembler to store the following code starting from address 2048

**dstart .equ 4000** // set variable dstart to 4000

**ld [a], %r1** // load content of address label "a" to register 1 (content is 15)

**ld [b], %r2** // load content of address label "b" to register 2 (content is 54)

**ld [c], %r3**  // load content of address label "c" to register 3 (content is -29)

**addcc %r2, %r3, %r0** // add contents of register 2 and 3 to register 0 (which is 0 because reg 0 is always 0). Also set the condition flags

**addcc %r1, %r0, %r4** // add contents of register 1 and 0 to register 4. Also set the condition flags

**subcc %r2, %r4, %r2** // content of register 2 subtract content of register 4 and store the result to register 2. Also set the condition flags

**st %r0, [a]** // store content of register 0 (which is 0) to memory location a

**st %r2, [y]** // store content of register 2 to memory location y

**jmpl %r15 + 4, %r0** // return from subroutine and go to the next instruction

**.org dstart**  // tells the assembler to store the following code starting from address "dstart" (which is set to 4000)

**a: 15** // store 15 (base 10) in address "a" (a is a label for address 4000)

**b: 54** // store 54 in address "b"

**c: -29** // store 15 in address "c"

**y: 0** // store 15 in address "y"

**.end** // end of the assembling process

// this program basically have 54 - 15 and store that result to memory location y.

// %r4 has the value 15 (hex f) after finish executing

! Program 3

**.begin** //start of the assembling process

**.org 2048** // tells the assembler to store the following code starting from address 2048

**main: ld [x], %r2** // load content of location x (which is 800) to register 2

**top: subcc %r1, 4, %r0** // subtract the value of register 1 by 4 and store result to register 0. Also set the condition flags

**be done** //if the z flag in psr is set to 1, jump to memory location "done". Else, go to the next instruction

**srl %r2, 1, %r2** // content of register 2 shift right logical by 3 bits (add 3 0's in msb), and store the result back to register 2

**addcc %r1, 1, %r1** // add content of register 1 by 1 and store result back to register 1. Also set the condition flags

**ba top** // jump to memory location of "top" for the next instruction

**done: st %r3, [y]** // store content of register 3 to memory location y

**jmpl %r15+4, %r0** // return from subroutine and go to the next instruction

**.org 3000** // tells the assembler to store the following code starting from address 3000

**x: 800** // store 800 in address "x"

**y: 0** // store 0 in address "y"

**.end** // end of the assembling process

// displacement value for ba top is - 4 words (-16 bytes), displacement value for be done is 4 words (16 bytes)

// %r1 has the value 4 after finish executing

! Program 4

**.begin** // start of the assembling process

**.org 2048** // set the next instruction to start from address 2048

**main: ld [a], %r1** // load content of address label "a" to register 1 (content is 0xa0) // Bits: 11 00001 000000 00000 1 0101110111000

**ld [b], %r2** // load content of address label "b" to register 2 (content is 0x33) // Bits: 11 00002 000000 00000 1 0101110111100

**ld [c], %r3** // load content of address label "c" to register 3 (content is 0x52) // Bits: 11 00003 000000 00000 1 0101111000000

**top: subcc %r4, 3, %r0** // subtract the value of register 1 by 3 and store result to register 0. Also set condition flags // Bits: 10 00000 010100 00100 1 0000000000011

**be done** //if the z flag in psr is set to 1, jump to memory location "done", else, go to the next instruction // Bits: 00 00001 010 0000000000000000001010

**subcc %r5, %r6, %r0** // subtract the value of register 5 by the value in register 6 and store result to register 0. Also set the condition flags // Bits: 10 00000 010100 00101 0 00000000 00110

**bneg else** // branch to address labelled "else" if the N flag in PSR is 1, otherwise go to the next instruction // Bits: 00 00110 010 0000000000000000000100

**orcc %r1, %r2, %r1** // do bitwise or with content of register 1 and 2, store the result in register 1. Also set the condition flags // Bits: 10 00001 010010 00001 0 00000000 00010

**addcc %r6, 1, %r6** // add the value of register 6 by 1 and store result to register 6. Also set the condition flags // Bits: 10 00110 010000 00110 1 0000000000001

**ba bottom** // jump to memory location of "bottom" for the next instruction // Bits: 00 01000 010 0000000000000000000011

**else: andcc %r1, %r3, %r1** // do bitwise and with content of register 1 and 3, store the result in register 1. Also set the condition flags // Bits: 10 00001 010001 00001 0 00000000 00011

**addcc %r5, 1, %r5** // add the value of register 5 by 1 and store result to register 5. Also set the condition flags // Bits: 10 00101 010000 00101 1 0000000000001

**bottom: addcc %r4, 1, %r4** // add the value of register 4 by 1 and store result to register 4. Also set the condition flags // Bits: 10 00100 010000 00100 1 0000000000100

**ba top** // jump to memory location of "top" for the next instruction // Bits: 00 01000 010 1111111111111111110110

**done: st %r3, [y]** // store the value in register 3 to memory location "y" // Bits: 11 00011 000100 00000 1 0101111000100

**jmpl %r15+4, %r0** // return from subroutine and go to the next instruction // Bits: 10 00000 111000 01111 1 0000000000100

**.org 3000** // tells the assembler to store the following code starting from address 3000

**a: 0xa0** // store 0xa0 in address "a" (3000)

**b: 0x33** // store 0x33 in address "b" (3004)

**c: 0x52** // store 0x52 in address "c" (3008)

**y: 2** // store 2 in address "y" (3012)

**.end** // end of the assembling process

// %r4 has the value 3 after finish executing