! Program 1

! Hint: op3 for subcc is 010100

.begin

.org 2048 Program starts saving lines in memory location 2048

dstart .equ 4000 dstart = location 4000

ld [a], %r1 Load a[4000] into register 1

ld [b], %r2 Load b [4004] into register 2

ld [c], %r3 Load c [4008] into register 3

addcc %r2, %r3, %r0 add value in register 2 to 3 and store it in register 0

addcc %r1, %r0, %r4 add value in register 1 to 0 and store it in register 4

subcc %r2, %r4, %r2 Subtract value in register 4 from 2 and store it in register 2

st %r0, [a] Store value in register 0 in [4000]

st %r2, [y] Store value in register 2 in [4012]

jmpl %r15 + 4, %r0 Return to routine

.org dstart

a: 15

b: 54

c: -29

y: 0

.end

Answer 15

11 00001 000000 00000 1 0111110100000

11 00010 000000 00000 1 0111110100100

11 00011 000000 00000 1 0111110101000

10 00000 010000 00010 0 0000000000011

10 00100 010000 00001 0 0000000000000

10 00010 010100 00010 0 0000000000100

11 00000 000100 00000 1 0111110100000

11 00000 000100 00010 1 0111110101100

10 00000 111000 01111 1 0000000000100

! Program 2

.begin

.org 2048

m\_start: ld [x], %r1 Load x into register 1

ld [y], %r2 load y into register 2

subcc %r1, %r2, %r0 subtract register 2 from 1 and save it in register 0

bneg else if above answer is negative branch to else

andcc %r1, %r2, %r3 add register 1 to 2 and save it in 3

ba done branch always to done

else: orcc %r1, %r2, %r3 Bitwise logical or register 1 and 2 and store it in 3

done: st %r3, [a] store value in register 3 in a

jmpl %r15+4, %r0 return to sub routine

x: 30

y: 48

a: 0

.end

Answer 62

11 00001 000000 00000 1 0100000100100

11 00010 000000 00000 1 0100000101000

10 00000 010100 00001 0 0000000000010

00 00110 010000 00000 0 0000000001100

10 00011 010001 00001 0 0000000000010

00 01000 010000 00000 0 0000000001000

10 00011 010010 00001 0 0000000000010

11 00000 000100 00011 1 0100000101100

10 00000 111000 01111 1 0000000000100

! Program 3

.begin

.org 2048

main: ld [x], %r2 load x into register 2

top: subcc %r1, 4, %r0 subtract 4 from register 1 and store it in register 0

be done branch if equal to done

srl %r2, 1, %r2 shift register 2 right by 1 and store in register 2

addcc %r1, 1, %r1 add 1 to register 1 and store in register 1

ba top branch always to top

done: st %r3, [y] store register 3 value in y

jmpl %r15+4, %r0 return to sub routine

.org 3000

x: 800

y: 0

.end

Answer = 4

11 00010 000000 00000 1 0101110111000

10 00000 010100 00001 1 0000000000100

00 00001 010000 00000 0 0000000010000

10 00010 100110 00010 1 0000000000001

10 00001 010000 00001 1 0000000000001

00 01000 010111 11111 1 1111111110000

11 00000 000100 00011 1 0101111000000

10 00000 111000 01111 1 0000000000100

! Program 4

.begin

.org 2048

main: ld [a], %r1 load a into register 1

ld [b], %r2 load b into register 2

ld [c], %r3 load c into register 3

top: subcc %r4, 3, %r0 substract 3 from register 4 and save it in registr 0

be done branch if equal to done

subcc %r5, %r6, %r0 substract 6 from register 5 and save it in registr 0

bneg else branch if negative to else

orcc %r1, %r2, %r1 bitwise logical or register 1 and register 2 and store in register 1

addcc %r6, 1, %r6 add 1 to register 6 and store in register 6

ba bottom branch always to bottom

else: andcc %r1, %r3, %r1 bitwise logical and register 1 and 3 and store in 1

addcc %r5, 1, %r5 add 1 to register 5 and store in register 5

bottom: addcc %r4, 1, %r4 add 1 to register 4 and store in register 4

ba top branch always to top

done: st %r3, [y] store register 3 in y

jmpl %r15+4, %r0 return to sub routine

.org 3000

a: 0xa0

b: 0x33

c: 0x52

y: 2

.end

Answer 3

11 00001 000000 00000 1 0101110111000

11 00010 000000 00000 1 0101110111100

11 00011 000000 00000 1 0101111000000

10 00000 010100 00100 1 0000000000011

00 00001 010000 00000 0 0000000101000

10 00000 010100 00101 0 0000000000110

00 00110 010000 00000 0 0000000010000

10 00001 010010 00001 0 0000000000010

10 00110 010000 01000 1 0000000000001

00 01000 010000 00000 0 0000000001100

10 00001 010001 00001 0 0000000000011

10 00101 010000 00101 1 0000000000001

10 00100 010000 01000 1 0000000000001

00 01000 010111 11111 1 1111111011000

11 00000 000100 00011 1 0101111000100

10 00000 111000 01111 1 0000000000100