Solutions for CSA H.W- 3 - GG2961

Q.1
a)
Load word from address A[10] into x5 (10 * 4 = 40 bytes)
lw x5, 40(x27)

b)
Store word from x5 into A[17] (17 * 4 = 68 bytes)
sw x5, 68(x27)

c)
Add x5 and x6, store result in x7
add x7, x5, x6

d)
add x3, x28, x29
addi x3, x3, 31
slli x3, x3, 2
lw x4, 0(x27, x3)
slli x6, x6, 2
sw x4, 0(x31, x6)

e)
i)
lw x4, 36(x30)
slli x4, x4, 2
lw x3, 0(x27, x4)
sub x5, x6, x3

ii)
lw x3, 32(x31)
lw x4, 16(x30)
add x3, x3, x4
slli x3, x3, 2
lw x4, 0(x27, x3)
sub x5, x6, x4

iii)
slli x3, x28, 1

```
slli x4, x3, 2
lw x5, 0(x30, x4)
slli x6, x28, 2
sw x5, 0(x31, x6)

addi x3, x3, 1
slli x4, x3, 2
lw x5, 0(x30, x4)
sw x5, 0(x27, x6)
```

iv)

```
addi x3, x28, -1
slli x3, x3, 2
lw x4, 0(x30, x3)
slli x4, x4, 2

addi x5, x28, 1
slli x5, x5, 2
lw x6, 0(x31, x5)
slli x6, x6, 2

add x3, x4, x6
sw x3, 0(x27, x28)
```

v)

```
lw x3, 16(x31)
lw x4, 48(x30)
add x3, x3, x4
slli x3, x3, 2
lw x4, 0(x27, x3)
sub x5, x6, x4
```

Q .2

(a) Sequence of instructions:
Code-

```
srli x7, x5, 16
addi x7, x7, -128
```

srai x7, x7, 2
and x7, x7, x6

i) srli x7, x5, 16

x5 = 0x00000000AAAAAAAA

Shifting x5 right logically by 16 bits:
x7 = 0x000000000000AAAA

ii) addi x7, x7, -128

x7 = 0x000000000000AAAA

Adding -128 to x7:
x7 = 0x000000000000AA2A

iii) srai x7, x7, 2

x7 = 0x000000000000AA2A

Arithmetic right shift by 2 bits:
x7 = 0x0000000000002A8A

iv) and x7, x7, x6

x7 = 0x0000000000002A8A

x6 = 0x1234567812345678

Performing bitwise AND between x7 and x6:
x7 = 0x0000000000000208

Final value of x7:
x7 = 0x0000000000000208

(b) Sequence of instructions:

Code-

slli x7, x6, 4

Steps:

slli x7, x6, 4

x6 = 0x1234567812345678

Shifting x6 left logically by 4 bits:
x7 = 0x2345678123456780

Final value of x7:
x7 = 0x2345678123456780

(c) Sequence of instructions:

Code-

srli x7, x5, 3

andi x7, x7, 0xFEF

Steps:

srli x7, x5, 3

x5 = 0x00000000AAAAAAAA

Shifting x5 right logically by 3 bits:
x7 = 0x0000000015555555

andi x7, x7, 0xFEF

x7 = 0x0000000015555555

Performing bitwise AND with 0xFEF:
x7 = 0x0000000000000545

Final value of x7:
x7 = 0x0000000000000545

Q. 3

Type, opcode, func3, func7, rs1, rs2, rd, imm: This will the flow of answer for each instruction and followed by hex number

add x5, x6, x7

R,0x33,0x0,0x0,6,7,5,-

0x007302B3

addi x8, x5, 512

I, 0x13, 0x0, -, 5, -, 8, 512

0x20028413

ld x3, 128(x27)

I, 0x3, 0x3, -, 27, -, 3, 128

0x080DB183

sd x3, 256(x28)

S, 0x23, 0x3, -, 28, 3, -,256

0x103E3023

beq x5, x6 ELSE

SB, 0x63, 0x0, -, 5, 6, -, 16

0x00628863

add x3, x0, x0

R, 0x33, 0x0, 0x0, 0, 0, 3, -

0x000001B3

auipc x3, FFEFA

U, 0x17, -, -, -, -, 3, FFEFA

0xFFEFA197

jal x3 ELSE

UJ, 0x6F, -, -, -, -, 3, 16

0x010001EF

Q. 4
a)

```
lw   x6, 0(x11)
slli x6, x6, 16
mv   x5, x6
```

b)

```
srli x5, x3, 7
andi x5, x5, 0x3F
li   x6, ~(0x3F << 23)
and  x4, x4, x6
slli x5, x5, 23
or   x4, x4, x5
```

x3 = 0

x4 = 0xffffffffffffffff

both x3 and x4 remain unchanged, and only bits 28 to 23 in x4 are replaced by bits 12 to 7 from x3.

c)

```
xori x5, x6, -1 # XOR x6 with -1 to invert all bits and store the result in x5
```

The result is the bitwise inversion of x6, which is stored in x5.

This single instruction achieves the desired result of the not pseudoinstruction.

Q. 5
a)

Offset Range:

$=[-220\times2,(220-1)\times2]$

$=[-2,097,152,2,097,150]$

PC Range:

If the current PC is 0x60000000, then:

Minimum possible PC value after the jump (for the largest negative offset) is:
0x60000000−2,097,152=0x5FE00000

Maximum possible PC value after the jump (for the largest positive offset) is:
0x60000000+2,097,150=0x601FFFFE

The range of addresses that can be reached using the jal instruction with the PC set to 0x60000000 is from:

0x5FE00000 to 0x601FFFFE

b)

The immediate can range from -2^{12} to 2^{12} - 1 (because the 13th bit is used for the sign).

Offset range=[−212×2,(212−1)×2]

=[−4096,4094]

Minimum possible PC value after the branch (for the largest negative offset) is:

0x60000000−4096=0x5FFFF000

Maximum possible PC value after the branch (for the largest positive offset) is:

0x60000000+4094=0x60000FFE

The range of addresses that can be reached using the beq instruction with the PC set to 0x60000000 is from:

0x5FFFF000 to 0x60000FFE

Q. 6

a) c code-

```
int acc = 0;
int i = 10;

while (i != 0) {
    i--;
    acc += 2;
}
```

b)

Total Instructions Executed:

In each of the N iterations: 4 instructions (beq, addi, addi, jal).

After the final iteration: The beq instruction runs one additional time to exit the loop.

Thus, the total number of instructions executed is:

Total instructions=N×4+1=4N+1

The total number of RISC-V instructions executed is:

4N+1

c)

C code-

```
int acc = 0;
int i = N;
```

```
while (i >= 0) {
    i--;
    acc += 2;
}
```

Q. 7

a)

Code-

```
 li      x7, 0

Outer_Loop:
    bge     x7, x5, End_Outer

    li      x29, 0

Inner_Loop:
    bge     x29, x6, End_Inner

    add     x11, x7, x29
    slli    x12, x29, 2
```

```
    add     x12, x10, x12
    sw      x11, 0(x12)

    addi    x29, x29, 1
    jal     x0, Inner_Loop

End_Inner:
    addi    x7, x7, 1
    jal     x0, Outer_Loop

End_Outer:
```

b)

Outer Loop Body:

Total: 2×10=202 \times 10 = 202×10=20 instructions.

Inner Loop Body:Total: 7×1×10=707 \times 1 \times 10 = 707×1×10=70 instructions.Post Inner Loop: 2 instructions per iteration, for 10 iterations.

Total: 2×10=202 \times 10 = 202×10=20 instructions.

Ans- 20+70+20=110 instructions executed.

Q. 8

a)

0x10000000: 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88

The value stored at 0x10000007 on a big-endian machine is 0x88

b)

Address: 0x10000000 0x10000001 0x10000002 0x10000003 0x10000004 0x10000005 0x10000006 0x10000007

Data: 0x88 0x77 0x66 0x55 0x44 0x33 0x22 0x11

The value stored at 0x10000007 on a little-endian machine is 0x1

Q. 9

lui x10, 0x12345

addi x10, x10, 0x678

slli x10, x10, 32

ori x10, x10, 0x12345678

Q.10

a)

The range of values for a 64-bit signed integer is:

-263 to $263-1$

$-9,223,372,036,854,775,808$ to $9,223,372,036,854,775,807$

For positive overflow:

$X6 \leq 9,223,372,036,854,775,679$

For negative overflow:

$X6 \geq -9,223,372,036,854,775,936$

The range of values for x6 that would result in no overflow is:

$-9,223,372,036,854,775,936 \leq x6 \leq 9,223,372,036,854,775,679$

b)

For positive overflow (when the result exceeds 9,223,372,036,854,775,807)

$128 - x6 \leq 9,223,372,036,854,775,807$

$X6 \geq -9,223,372,036,854,775,679$

For negative overflow (when the result is less than -9,223,372,036,854,775,808)

$128 - x6 \geq -9,223,372,036,854,775,808$

$X6 \leq 9,223,372,036,854,775,936$

The range of values for x6 that would result in no overflow is:

$-9,223,372,036,854,775,936 \leq x6 \leq 9,223,372,036,854,775,679$

c)

For positive overflow (when the result exceeds 9,223,372,036,854,775,807)

$X6 - 128 \leq 9,223,372,036,854,775,807$

$X6 \leq 9,223,372,036,854,775,935$

For negative overflow (when the result is less than -9,223,372,036,854,775,808)

$X6 \geq -9,223,372,036,854,775,680$

The range of values for x6 that would result in no overflow is:

$-9,223,372,036,854,775,680 \leq x6 \leq 9,223,372,036,854,775,935$