EAST WEST UNIVERSITY

Department of Computer Science and Engineering

Course Code: CSE360

Course Title: Computer Architecture

Section: 03

Semester: Spring 2024

Assignment 02

<u>Submitted to:</u>

Md. Ezharul Islam, Phd

Professor

Department of Computer Science and Engineering

<u>Submitted by:</u>

Name: B M Shahria Alam

ID: 2021-3-60-016

Chapter-2

2.1

f = g+ (h-5);

```
addi  x5, x7, -5
add   x5, x5, x6
```

[addi f, h, -5 (note, no subi) add, f, f, g]

2.2

```
add  f, g, h
add  f, i, f
```

∴ f = g+h+i

2.3

```
Sub   x30, x28, x29   // compute i-j
Slli  x30, x30, 3     // multiply by 8 to convert the
                         word offset to a byte
                         offset
ld    x30, 0(x3)
Sd    x30, 64(x11)    // store in B[8]
```

2.4

B[g] = A[f] + A[f+1]

```
slli  x30, x5, 3      // x30 = f×8
add   x30, x10, x30   // x30 = &A[f]
slli  x31, x6, 3      // x31 = g×8
add   x31, x11, x31
```

```
ld    x5, 0(x30)
addi  x12, x30, 8
ld    x30, 0(x,12)
add   x30, x30, x5
sd    x30, 0(x31)
```

2.7

```
slli   x28, x28, 3    // x28 = i×8
ld     x28, 0(x10)    // x28 = A[i]
slli   x29, x29,3     // x29 = j×8
ld     x29, 0(x11)    // x29 = B[j]
add    x29, x28,x29
sd     x29, 64(x11)   //store result in B[8]
```

2.8.

f = 2 × (&A)

```
addi   x30, x10, 8    // x30 = &A[1]
addi   x31, x10, 0    // x31 = &A
sd     x31, 0(x30)    // ΣA[] = &A
ld     x30, 0(x30)    // x30 = A[1] = &A
add    x5, x30, x31   // f = &A + &A
```

2.9

	type	opcode funct 3,7	rs1	rs2	rd	imm
addi x30, x10, 8	I-type	0x13, 0x0	10	–	30	8
addi x31, x10, 0	R-type	0x13, 0x0	10	–	31	0
sd x31, 0(x30)	S-type	0x23, 0x3	31	30	–	0
ld x30, 0(x30)	I-type	0x3, 0x3	30	–	30	0
add x5, x30, x31	R-typ	0x33, 0x	30	31	5	–

2.11

There is an overflow if $128 + x6 > 2^{63} - 1$

In other words, if $x6 > 2^{63} - 129$

There is also an overflow if $128 + x6 < -2^{63}$.

In other words, if $x6 < -2^{63} - 128$ (which is impossible given the range of x6)

2.11.3

There is an overflow if $x6 - 128 > 2^{63} - 1$

In other word, if $x6 < 2^{63} + 128$ (which is impossible given the rang of x6)

There is also an overflow if $x6 - 128 < -2^{63}$

In other words, if $x6 < -2^{63} + 128$

2.18

It can be done in eight RISC-V instructions:

```
addi x7, x0, 0x3f // Create bit mask
slli  x7, x7, 11   // shift the masked bits
and  x28, x5, x7   // Apply the mask to x5.
slli x7, x6, 15    // shift the mask
xori x7, x7, -1    // This is a NOT operation
and  x6, x6, x7    // "Zero out"
slli x28, x28, 15  // move selection from x5
or   x6, x6, x28 // load bite.
```

2.23

```
loop:
    addi  x29, x29, -1 // subtract 1 from x29
    bgt   x29, x0, loop // continue if x29 not
    addi  x29, x29, 1 // add back
```

2.24

```
   acc = 0
     i = 10;
  while (i != 0) {
     acc += 2
     i--;
  }
```

2.25

```
Loop I ;
    addi  x7,  x0, 0   // init  i=0
    bge   x7, x5, EnD I // while i<a
    addi  x29, x0, 0    // Init j=0

LOOP J;
    bge  x29 , x6, END J // while j<b
    add  x31 , x7, x29  // x31 = i+j
    sd   x31 , 0(x30)    // D[4xj] = x31
    addi x30, x30, 32 // x30 = &D[4x(j+1)]
    jal  x0 , LOOP J
END J:
    addi x7 , x7, 1  // i++;
    jal  x0, LOOP I
ENDI
```

2.27

```
int i;
bor (i=0; i<100; i++) {
    result += * MemArray
    MemArray++;
}
result result;
```

```
int i;
for (i=0; i<100; i++){
    result += Mem Array [i];
  }
return result;
```

2.31

```
addi   x2,  x2,  -16      // Allocate stack space
sd     x1,  0(x2)
add    x5,  x12, x13      // x5 = c+d
sd     x5,  8(x2)         // save c+d on the stack
jal    x1,  g             / call x10 = g (a,b)
ld     x11, 8(x2)         // Reload
jal    x1,  g
ld     x1,  0(x2)         // Restore
addi   x2,  x2,  16       / Restore
jalr   x0,  x1
```

2.36

```
lui    x 10 ,  0x11
addi   x 10, x10
slli   x10, x10, 32
addi   x5,  x5
add    x10, x10, x5
```

2.37

Set max:

```
try:
    lr.d   x5, (x10)     // load-reserve * shvar
    bge    x5, x11, release
    addi   x5, x11, 0
release:
    sc.d   x7, x5
    bne    x7, x0, try
    jalr   x0, x1
```

2.40.1

Take the weight average

= 0.7×2 + 0.1×6 + 0.2×3

= 2.6

2.41

```
ldr    x28, x5(x10), 3    // Load x28 = A[f]
addi   x5, x5, 1          // f++
ldr    x29, x5(x10), 3    // load x29 = A[f+1]
add    x29, x29, x28      // add x29 = A[f] +
                                       A[f+1]
sdr    x12, x6(x11), 3    // store B[g] = x29
```

2.42

```
ldr   x 28 , x28 , (x 10) , 3 // Load   x28=A[i]
ldr  x29 , x29 , (x11) , 3 // Load
add   x29, x28, x29
sd    x29   , 64 (x11)   //store B[8] =x29
```