Zuneera Abdul Aziz
BSSE23058
Section A.

Date: Day.

Assignment # 1.
COAL.

Q1

X IC = 8.0×10^8 Execution time / CPU time = 0.9s
Y IC = 1.0×10^9 Execution time = 1.2s
} Clock rate = 2 GHz

a. CPI = ?

$$\text{Clock cycle time} = \frac{1}{\text{Clock Rate}} = \frac{1}{2 \times 10^9} = 5 \times 10^{-10}\ s$$

Execution time / CPU time = CPU clock cycles × Clock cycle time.

For X compiler

$$0.9 = \text{CPU clock cycles} \times 5 \times 10^{-10}$$

$$\frac{0.9}{5 \times 10^{-10}} = \text{CPU clock cycles}$$

$$1.8 \times 10^9 = \text{CPU clock cycles.}$$

$$CPI = \frac{\text{CPU clock cycles}}{IC}$$

$$= \frac{1.8 \times 10^9}{8.0 \times 10^8}$$

$$\boxed{CPI_X = 2.25}$$

For Y compiler.

Execution time = CPU clock cycles × clock cycle time.

$$1.2 = \text{CPU clock cycles} \times 5 \times 10^{-10}$$

$$\frac{1.2}{5 \times 10^{-10}} =$$

$$2.4 \times 10^9 = \text{CPU clock cycles}$$

$$CPI = \frac{\text{CPU clock cycles}}{IC}$$

$$CPI = \frac{2.4 \times 10^9}{1.0 \times 10^9} = 2.4$$

Date: Day:

b. Execution time is same. $CPU\ time_X = CPU\ time_Y$

$$CPI = \frac{CPU\ clock\ cycles}{IC}$$

$$CPI \times IC = CPU\ clock\ cycles$$

$$CPI \times IC = Execution\ time / CPU\ time$$

$$CPI \times IC = CPU\ time \div \frac{1}{Clock\ Rate}$$

$$CPI_X \times IC = CPU\ time \times Clock\ Rate$$

$$\frac{CPI_X \times IC}{CPU\ time} = Clock\ Rate$$

$$\frac{Clock\ Rate_X}{Clock\ Rate_Y} = \frac{\frac{CPI_X \times IC_X}{CPU\ time_X}}{\frac{CPI_Y \times IC_Y}{CPU\ time_Y}}$$

Since CPU time is same, cancels each other

$$\frac{Clock\ Rate_X}{Clock\ Rate_Y} = \frac{CPI_X \times IC_X}{CPI_Y \times IC_Y}$$

$$= \frac{2.25 \times 8.0 \times 10^8}{2.4 \times 1.0 \times 10^9}$$

$$= 0.75$$

Clock rate of compiler X is 0.75 time Clock Rate of Y.

Processor X is 25% slower than Processor Y.

Processor Y is faster by 33% than Processor X.

c. Z $IC = 6 \times 10^8$ $CPI = 1.0$

Execution time$_Z$ = ?

$$CPI = \frac{CPU\ clock\ cycles}{IC}$$

$$1 \times 6 \times 10^8 = CPU\ clock\ cycles$$

6×10^8

Execution time/CPU$_Z$ = CPU clock cycles × clock cycle time

$= 6 \times 10^9 \times \frac{1}{2 \times 10^9}$

Execution time$_Z$ = 0.3 s

Execution time$_Y$ = 0.9 s

Execution time$_X$ = 1.2 s

Speedup = $\frac{\text{Execution time old}}{\text{Execution time new}}$

Speedup using X = $\frac{0.9}{0.3}$ Old → Y, New → Z.

= 3 times faster

Speedup using Y = $\frac{\text{Execution time Y}}{\text{Execution time Z}}$

$= \frac{1.2}{0.3}$

= 4 times faster.

d. Compiler Z is 3 times faster than Compiler X. Compiler Z runs 4 times faster than Compiler Y. Compiler Z is the best because it requires fewer instructions and has a lower CPI, making execution much faster. Compiler Y is the slowest since Compiler Z improves performance more (4×) compared to X (3×).

Date: Day:

Q2.

Total CPU time = IC × CPI × CCT

= IC × CPI × (CCT) → $\frac{1}{\text{Clock frequency}}$

For Arithmetic Instruction.

$\text{CPU time}_{\text{Arithmetic}} = 3.0\times10^9 \times 2 \times \frac{1}{2.5\times10^9}$

$= 2.4\text{s}$

For Load/Store Instructions

$\text{CPU time}_{\text{Load/Store}} = 1.5\times10^9 \times 10 \times \frac{1}{2.5\times10^9}$

$= 6\text{s}$.

For Branch Instructions

$\text{CPU time}_{\text{Branch}} = 3.0\times10^8 \times 4 \times \frac{1}{2.5\times10^9}$

$= 0.48\text{s}$

1 processor. Total CPU time = 2.4 + 6 + 0.48

$= 8.88\text{s}$ ⇒ time for 1 processor.

For Arthmetic & load/store Instruction per second.

divided by 2.8 × p.

Branch instructions remain same. per second.

2 processors in parallel. / multiple cores

$IPS_{A_{new}} = \frac{IPS_A}{0.8\sqrt{p}}$ | $IPS_{\text{Load/Store}_{new}} = \frac{IPS_{\text{Load/Store}}}{0.8\sqrt{p}}$ | $IPS_{B_{new}} = IPS_B$ same.

$IPS_{A_{new}} = \frac{3.0\times10^9}{0.8\times\sqrt{2}}$ | $IPS_{\text{Ls new}} = \frac{1.5\times10^9}{0.8\times\sqrt{2}}$ | $IPS_{B_{new}} = 3.0\times10^8$

$= 2.65\times10^9$. | $= 1.33\times10^9$

$\text{CPU time} = 2.65\times10^9 \times 2 \times \frac{1}{2.5\times10^9}$ | $\text{CPU time}_{\text{Load/Store}} = 1.33\times10^9 \times 10 \times \frac{1}{2.5\times10^9}$

$\text{CPU time} = 3.0\times10^8 \times 4 \times \frac{1}{2.5\times10^9}$

Date: Day:

= 2.12s | = 5.32s | = 0.48s

Total CPU time 2 processors = 2.12 + 5.32 + 0.48 = 7.92s

for 4 processors

New IPS.

$IPS_{A\,new} = \frac{3.0 \times 10^9}{0.8 \times \sqrt{4}}$ | $IPS_{4S\,new} = \frac{1.5 \times 10^9}{0.8 \times 54}$ | $IPS_{B\,new} = 3.0 \times 10^9$

$= 1.88 \times 10^9$ | $= 9.38 \times 10^8$

$CPU\ time_A = 1.88 \times 10^9 \times 2 \times \frac{1}{2.5 \times 10^9}$ | $CPU\ time_{4S} = 9.38 \times 10^8 \times 10 \times \frac{1}{2.5 \times 10^9}$ | $CPU\ time_B = 3.0 \times 10^9 \times 4 \times \frac{1}{2.5 \times 10^9}$

$= 1.504\ s$ | $= 3.752\ s$ | $= 0.48\ s$

Total CPU time 4 processors = 1.504 + 3.752 + 0.48 = 5.736s ≈ 5.74s

for 8 processors

$IPS_{A\,new} = 1.327 \times 10^9$ | $IPS_{4S\,new} = 0.663 \times 10^9$ | $IPS_{B} = 3.0 \times 10^9$

$CPU\ time_A = 1.06\ s$ | $CPU\ time_{4S} = 2.652s$ | $CPU\ time_B = 0.48s$

Total CPU time 8 processors = 1.06 + 2.652 + 0.48 = 4.19s.

b. Speedup = $\frac{\text{Execution time old}}{\text{Execution time new}}$

Execution time old = 1 processors execution time = 8.88s.

Speedup relative to 2 processors = $\frac{8.88}{7.92}$ = 1.12 times faster.

Date:

Speedup relative to 4 processors $= \frac{8.88}{5.74}$

$= 1.55$ times.

Speedup relative to 8 processors $= \frac{8.88}{4.19}$

$= 2.12$ times faster.

c. For 1 processor.

CPU time Arithmetic $= 3.0 \times 10^9 \times 4 \times \frac{1}{2.5 \times 10^9}$

$= 4.8s$

Total New Execution time 1 processor $= 4.8 + 6 + 0.48$

$= 11.28s$

For 2 processors.

CPU time Arithmetic $= 2.65 \times 10^9 \times 4 \times \frac{1}{2.5 \times 10^9}$

$= 4.24s$

Total New Execution time 2 processors $= 4.24 + 5.32 + 0.48$

$= 10.04s$

For 4 processors.

CPU time Arithmetic $= 1.88 \times 10^9 \times 4 \times \frac{1}{2.5 \times 10^9}$

$= 3.008$

Total New Execution time 4 processors $= 3.008 + 3.752 + 0.48$

$= 7.24s$

For 8 processors

CPU time Arithmetic $= 1.327 \times 10^9 \times 4 \times \frac{1}{2.5 \times 10^9}$

$= 2.12s$

Total Execution time 8 processors $= 2.12 + 2.652 + 0.48$

$= 5.24s$

Date: _____ Day: _____

	old		New
1 processor	8.88 s	→	11.28 s
2 processors	7.92 s	→	10.04 s
4 processors	5.74 s	→	7.24 s
8 processors	4.19 s	→	5.26 s

Execution time increased for all the processors as doubling CPI for Arithmetic instructions leads to more cycles.

d. Execution time of 4 processors = 5.74 s
Execution time of 1 processors = 8.88 s.] = ?
Execution time 1 processor = Execution time 4 processors
CPU time (Arithmetic) + CPU time (Load/Store) + CPU time (Branch) = 5.74

$$2.4 + 1.5\times10^9 \times CPI_{new} \times \frac{1}{2.5\times10^9} + 0.48 = 5.74$$

$$2.4 + 0.6\,CPI_{new} + 0.48 = 5.74$$

$$2.88 + 0.6\,CPI_{new} = 5.74$$

$$CPI_{new} = \frac{5.74 - 2.88}{0.6}$$

$$= \frac{2.86}{0.6}$$

$$= 4.76$$

10 → 4.76 CPI

Q3 a. speedup = 2 times faster.

$$\text{speedup} = \frac{\text{En time old}}{\text{En time new}}$$

$$\text{En time}_{old} = \text{CPU time}_{(FP)} + \text{CPU time}_{(INT)} + \text{CPU time}_{(LS)} + \text{CPU time}_{(Branch)}$$

Date: Day:

$$= \left(60\times10^6 \times 2 \times \frac{1}{2\times10^9}\right) + \left(120\times10^6 \times 1 \times \frac{1}{2\times10^9}\right) + \left(90\times10^6 \times 5 \times \frac{1}{2\times10^9}\right) + \left(18\times10^6 \times 3 \times \frac{1}{2\times10^9}\right)$$

$$= 0.06 + 0.06 + 0.225 + 0.027$$

$$\text{old}_{\text{dw}} = 0.372$$

$$\text{speed up} = \frac{\text{Ent time old}}{\text{Ent time new}}$$

$$2 = \frac{0.372}{60\times10^6 \times \frac{1}{2.0\times10^9} \times CPI + 0.06 + 0.225 + 0.027}$$

$$0.03CPI + 0.312 = \frac{0.372}{2}$$

$$CPI = \frac{0.186 - 0.312}{0.03}$$

$$= -4.2$$

A negative CPI isn't possible, so improving the CPI of FP instructions isn't entirely on it, maybe. In order to achieve better speed up., other components may need to improve as well.

b. $\text{speedup} = \frac{\text{Ent time old}}{\text{Ent time new}}$

$$2 = \frac{0.372}{0.06 + 0.06 + 0.045\,CPI + 0.027}$$

$$0.147 + 0.045[PI = 0.186$$

$$CPI = \frac{0.186 - 0.147}{0.045}$$

$$= 0.867$$

reducing from 5 → 0.867 CPI, achieves the speed up.

Date: Day.

So reducing L/S CPI is much more effective than reducing the CPI of FP for achieving the speedup required.

c. FP new CPI $= \left(1 - \frac{40}{100}\right) \times 2.$

$= 0.6 \times 2$

$= 1.2$

INT new CPI $= \left(1 - \frac{40}{100}\right) \times 1$

$= 0.6$

Branch new CPI $= \left(1 - \frac{30}{100}\right) \times 3$

$= 0.7 \times 3$

$= 2.1$

L/S new CPI $= \left(1 - \frac{30}{100}\right) \times 5$

$= 0.7 \times 5$

$= 3.5$

New Ex time $= \left(1.2 \times 60 \times 10^6 \times \frac{1}{2 \times 10^9}\right) + \left(120 \times 10^6 \times 0.6 \times \frac{1}{2 \times 10^9}\right) +$

$\left(3.5 \times 90 \times 10^6 \times \frac{1}{2 \times 10^9}\right) + \left(2.1 \times 18 \times 10^6 \times \frac{1}{2 \times 10^9}\right)$

$= 0.036 + 0.036 + 0.1575 + 0.0189$

New Ex time $= 0.2484$

Speedup $= \frac{\text{Ex time old}}{\text{Ex time new}}$

$= \frac{0.372}{0.2484}$

$= 1.498$

$= 1.5$

Date: _____ CODE ASSEMBLY CODE Day: Explan...

Q4

```
int main (){
    int x5 = 10;
    int x6 = 5;
    int x7;
    if (x5 > x6){
        x7 = 1;
    }else{
        x7 = 0;
    }
    return 0;
}
```

```
main:
    bgt x5, x6, one
    addi x7, x0, 0
    jal end
one:
    addi x7, x0, 1
end:
```

Compare x5 & x6 if x5 > x6, jump to one

add immediate x0 & 0 and store in x7 that is 0

jump to end if okay

add x0 & 1 and store it in x7.

Q5

```
int main (){
    int x8[] = {1, 2, 3, 4, 5};
    int x9 = sizeof(x8) / sizeof(x8[0]);
    int x10 = 0; // Initialize sum.
    for(int x11 = 0; x11 < x9; x11++){
        x10 += x8[x11];
    }
    return 0;
}
```

```
main:
    add x10, x0, x0      // x10 = 0.
    add x11, x0, x0      // x11 = 0  // index i = 0
loop:
    bge x11, x9, exit    // if (x11 >= x9) then exit loop // bge branch greater
    slli x12, x11, 2     // shift left logical immediate
                         // integer takes 4 bytes so we multiply by 2 which is taken in power of 2 as 2^2 = 4.
                         // x12 = x11 * 4  (i * 4) // getting byte offset
    add x12, x12, x8     // Address of arr[i] = base + offset byte.
                         // [x8]  [x12]
                         // x12 = x12 + x8
```

Date: Day:

```
lw   x13, 0(x12)   // loading word from memory into
add  x10, x10, x13 // x10 += x13. // sum += a[i];
addi x11, x11, 1   // x11++ // i++.
j    loop          // back to loop / repeat.
exit:              // if condition fails loop ends.
```

Q6

```
int main(){
    int A[3][3] = ---- ;
    int B[3][3] = ---- ;
    int C[3][3] = {0};
    for(int i=0; i<3; i++){
        for(int j=0; j<3; j++){
            C[i][j]=0
            for(int k=0; k<3; k++){
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    return 0;
}
```

```
main:
    li x23, 0      // i=0 first loop.
jloop:
    li x24, 0      // j=0 . second loop.
kloop:
    li x25, 0      // k=0 third loop
    li x26, 0      // C[i][j]=0.
    li x27, 0      // k=first element
    li x28, 0      // A[i][k]
    li x29, 0      // B[k][j]
    mul x28, x23, 3 // x28 = i*3 first loop A row offset.
    add x28, x28, x27 // x28 = i*3 + k total A index.
```

Date: Day:

```
slli  x28, x28, 2     // getting the offset byte of word 2^2 = 4
add   x28, x28, x20   // base address of A.
lw    x30, 0(x28)     // Load A[i][k] into x30.
```

B

```
mul  x29, x27, 3     // x29 = k*3
add  x29, x29, x24   // x29 = k*3 + j
slli x29, x29, 2     // 2^2 = 4
add  x29, x29, x21   // Base address of B
lw   x31, 0(x29)     // Load B(k)(j) into x31
```

— send to C —

```
mul x32, x30, x31   // C = A + B
add x26, x26, x32   // result in C.
```

— store result to C. —

```
mul x28, x23, 3     // x28 = i*3
add x28, x28, x24   // x28 = i*3 + j
slli x28, x28, 2
add x28, x28, x22
sw  x26, 0(x28)     // store to memory.
```

— increment the loops ~~inner~~

```
addi x24, x24, 1    // j++
blt  x24, 3, kloop.
```

— increment outer loop —

```
addi x23, x23, 1    // i++
blt  x23, 3, jloop.
```

end. ~~x10, 10~~

Q7

```
for(int i=0; i < size-1; i++){
    for(int j= i+1; j < size; j++){
        if (arr[j] > arr[j+1]){
            swap(arr[i], arr[j])
        }
    }
```

Day:

```
main:
    li x5, 0.        // i=0
outerloop:
    sub x6, x28, x5  // size-i
    addi x6, x6, -1  // size-i-1
    bge x5, x6, exit // if (i>=size-1), exit loop
    addi x7, x5, 1   // j=i+1
innerloop:
    bge x7, x28, incrementi  // if j>=size go to incrementi
    slli x8, x5, 3   // 2^3=8.  i*8.
    slli x9, x7, 3   // 2^3=8   i*8
    add x8, x8, x27  // address of arr[i]
    add x9, x9, x27  // address of arr[j]
    ld x10, 0(x8)    // Load arr[i] into x10
    ld x11, 0(x9)    // Load arr[j] into x11
    ble x10, x11, false  // less than or equal // if (arr[i]<=arr[j]) no swap
    sd x10, 0(x9)    // store arr[i] in arr[j]
    sd x11, 0(x8)    // store arr[j] into arr[i]
false:
    addi x7, x7, 1   // j++
    j innerloop      // jump back to innerloop
incrementi:
    addi x5, x5, 1   // i++
    j outerloop
exit:
```