Homework 1 Solution

1.5: a. (6 Points)

Performance of P1 (instructions/sec) = $3 \times 10^{9}/1.5 = 2 \times 10^{9}$
Performance of P2 (instructions/sec) = $2.5 \times 10^{9}/1.0 = 2.5 \times 10^{9}$
Performance of P3 (instructions/sec) = $4 \times 10^{9}/2.2 = 1.8 \times 10^{9}$

P2 has the highest performance.

1.5: b (6 Points)

Cycles (P1) = $10 \times 3 \times 10^{9} = 30 \times 10^{9}$ s
Cycles (P2) = $10 \times 2.5 \times 10^{9} = 25 \times 10^{9}$ s
Cycles (P3) = $10 \times 4 \times 10^{9} = 40 \times 10^{9}$ s

No. instructions (P1) = $30 \times 10^{9}/1.5 = 20 \times 10^{9}$
No. instructions (P2) = $25 \times 10^{9}/1 = 25 \times 10^{9}$
No. instructions (P3) = $40 \times 10^{9}/2.2 = 18.18 \times 10^{9}$

1.5: c (6 Points)

CPI new = CPI old × 1.2, then CPI (P1) = 1.8, CPI(P2) = 1.2, CPI(P3) = 2.6
f = No. instr. × CPI/time, then
f (P1) = $20 \times 10^{9} \times 1.8/7 = 5.14$ GHz
f (P2) = $25 \times 10^{9} \times 1.2/7 = 4.28$ GHz
f (P1) = $18.18 \times 10^{9} \times 2.6/7 = 6.75$ GHz

1.6: a (6 Points)

Class A: 10^{5} instr. Class B: 2×10^{5} instr. Class C: 5×10^{5} instr. Class D: 2×10^{5} instr.
Time = No. instr. × CPI/clock rate
Total time P1 = $(10^{5} + 2 \times 10^{5} \times 2 + 5 \times 10^{5} \times 3 + 2 \times 10^{5} \times 3)/(2.5 \times 10^{9}) = 10.4 \times 10^{-4}$ s
Total time P2 = $(10^{5} \times 2 + 2 \times 10^{5} \times 2 + 5 \times 10^{5} \times 2 + 2 \times 10^{5} \times 2)/(3 \times 10^{9}) = 6.66 \times 10^{-4}$ s
CPI (P1) = $10.4 \times 10^{-4} \times 2.5 \times 10^{9}/10^{6} = 2.6$
CPI(P2) = $6.66 \times 10^{-4} \times 3 \times 10^{9}/10^{6} = 2.0$

1.6: b (6 Points)
Clock cycles (P1) = $10^{5} \times 1 + 2 \times 10^{5} \times 2 + 5 \times 10^{5} \times 3 + 2 \times 10^{5} \times 3 = 26 \times 10^{5}$
Clock cycles (P2) = $10^{5} \times 2 + 2 \times 10^{5} \times 2 + 5 \times 10^{5} \times 2 + 2 \times 10^{5} \times 2 = 20 \times 10^{5}$

1.7: a (6 Points)
CPI = $T_{exec} \times f$/No. instr.
Compiler A CPI = 1.1

Compiler B CPI = 1.25

1.7: b (6 Points)

f_B/f_A = (No. instr.(B) × CPI(B))/(No. instr.(A) × CPI(A)) = 1.37

1.7: c (6 Points)
$T_A/T_{new} = 1.67$

$T_B/T_{new} = 2.27$

1.8.1 (6 Points)
$C = 2 \times DP/(V^2 \times F)$
Pentium 4: C = 3.2E–8F

Core i5 Ivy Bridge: C = 2.9E–8F

1.8.2 (6 Points)
Pentium 4: 10/100 = 10%

Core i5 Ivy Bridge: 30/70 = 42.9%

1.8.3 (6 Points)
$(S_{new} + D_{new})/(S_{old} + D_{old}) = 0.90$
$D_{new} = C \times V_{new}^2 \times F$
$S_{old} = V_{old} \times I$
$S_{new} = V_{new} \times I$
Therefore:
$V_{new} = [D_{new}/(C \times F)]^{1/2}$
$D_{new} = 0.90 \times (S_{old} + D_{old}) - S_{new}$
$S_{new} = V_{new} \times (S_{old}/V_{old})$
Pentium 4:
$S_{new} = V_{new} \times (10/1.25) = V_{new} \times 8$
$D_{new} = 0.90 \times 100 - V_{new} \times 8 = 90 - V_{new} \times 8$
$V_{new} = [(90 - V_{new} \times 8)/(3.2E8 \times 3.6E9)]^{1/2}$
$V_{new} = 0.85$ V
Core i5:
$S_{new} = V_{new} \times (30/0.9) = V_{new} \times 33.3$
$D_{new} = 0.90 \times 70 - V_{new} \times 33.3 = 63 - V_{new} \times 33.3$
$V_{new} = [(63 - V_{new} \times 33.3)/(2.9E8 \times 3.4E9)]^{1/2}$

$V_{new} = 0.64$ V

1.10.1 (6 Points)
Die area 15cm = wafer area/dies per wafer = $\pi \times 7.5^2/84 = 2.10$ cm^2
Yield 15cm = $1/(1 + (0.020 \times 2.10/2))^2 = 0.9593$
Die area 20cm = wafer area/dies per wafer = $\pi \times 10^2/100 = 3.14$ cm^2

Yield 20cm = $1/(1 + (0.031 \times 3.14/2))^2 = 0.9093$

1.10.2 (6 Points)
Cost/die 15cm = 12/(84 × 0.9593) = 0.1489

Cost/die 20cm = 15/(100 × 0.9093) = 0.1650

1.10.3 (6 Points)
Die area 15cm = wafer area/dies per wafer = $\pi \times 7.5^2/(84 \times 1.1) = 1.91$ cm^2
Yield 15cm = $1/(1 + (0.020 \times 1.15 \times 1.91/2))^2 = 0.9575$
Die area 20cm = wafer area/dies per wafer = $\pi \times 10^2/(100 \times 1.1) = 2.86$ cm^2
Yield 20cm = $1/(1 + (0.03 \times 1.15 \times 2.86/2))^2 = 0.9082$

Homework 2 Solution

2.1 (7 Points)

```
ADDI X3, X2, -5

ADD X0, X1, X3
```

2.3 (7 Points)

```
SUB X9, X3, X4          // compute i-j
LSL X9, X9, #3          // convert the word offset to a byte offset: X9= X9*8
ADD X11, X6, X9         // compute address of A[i-j]
LDUR X10, [X11, #0]     // load A[i-j]
STUR X10, [X7, #64]     // store in B[8]
```

2.4 (12 Points)

```
B [g] = A[f] + A[f+1]
LSL X9, X0, #3 // X9 = f*8
ADD X9, X6, X9 // X9 = &A[f]
LSL X10, X1, #3 // X10 = g*8
ADD X10, X7, X10 // X10 = &B[g]
LDUR X0, [X9, #0] // f = A[f]
ADDI X11, X9, #8 // (*) X11 = X9 + 8 (i.e., X11 = &A[f+1])
LDUR X9, [X11, #0] // X9 = A[f+1]
ADD X9, X9, X0 // X9 = X9 + f (i.e., x9 = A[f+1] + A[f])
STUR X9, [X10, #0] // B[g] = X9 (i.e., B[g] = A[f+1] + A[f])
```

2 .5 (7 Points)

```
LSL X9, X0, #3 // X9 = f*8
ADD X9, X6, X9 // X9 = &A[f]
LSL X10, X1, #3 // X10 = g*8
ADD X10, X7, X10 // X10 = &B[g]
LDUR X0, [X9, #0] // f = A[f]
LDUR X9, [X9, #8] // X9 = A[f+1]
ADD X9, X9, X0 // X9 = X9 + f (i.e., X9 = A[f+1] + A[f])
STUR X9, [X10, #0] // B[g] = X9 (i.e., B[g] = A[f+1] + A[f])
```

2.9 (7 Points)

```
ADDI X9, X6, #8        // X9 = address of A[0] +8 = &A[1] (in the Language
of C++ or * in C)
ADD X10, X6, XZR    // X10 =  address of A[0] =&A[0]
STUR X10, [X9, #0]  // A[1] = &A[0]
LDUR X9, [X9, #0]    // X9 = A[1]
ADD X0, X9, X10      // f = &A[0] + &A[0]
```

or f=(&A[0]) +(&[0]A) or simply f = 2*(&A[0])

2.10 (18 Points)

Instruction	**Type**	**Opcode**	**rm**	**rn**	**Rd/rt**	**Immed/Address**
ADDI X9, X6, #8	I-type	580/0x244 1001000100		6 00110	9 01001	8 000000001000
ADD X10, X6, XZR	R-type	1112/0x458 10001011000	31 11111	6 00110	10 01010	
STUR X10, [X9, #0]	D-type	1984/0x7c0 11111000000		9 01001	10 01010	0 000000000
LDUR X9, [X9, #0]	D-type	1986/0x7c2 11111000010		9 01001	9 01001	0 000000000
ADD X0, X9, X10	R-type	1112/0x458 10001011000	10 01010	9 01001	0 00000	

The answer shows in Decimal and Binary separately.

2.22 (7 Points)

X 1 = 2

2.25.1 (7 Points)

The final value of X0 is 20.

2.41.1 (7 Points)

No. The resulting machine would be slower overall.

Current CPU requires (num arithmetic * 1 cycle) + (num load/store * 10cycles) + (num branch/jump * 3 cycles) = 500*1 + 300*10 + 100*3 = 3800 cycles.

The new CPU requires (0.75*num arithmetic * 1 cycle) + (num load/store * 10 cycles) + (num branch/jump * 3 cycles) = 375*1 + 300*10 + 100*3 = 3675 cycles.

However, given that each of the new CPU's cycles is 10% longer than the original CPU's cycles, the new CPU's 3675 cycles will take as long as 4042.5 cycles on the original CPU.

2.41.2 (7 Points)

If we double the performance of arithmetic instructions by reducing their CPI to 0.5, then the CPU will run the reference program in (500*0.5)+(300*10) + 100*3 = 3550 cycles. This represents a speedup of 1.07.
If we improve the performance of arithmetic instructions by a factor of 10 (reducing their CPI to 0.1), then the CPU will run the reference program in (500*0.1) + (300*10) + 100*3 = 3350 cycles. This represents a speedup of 1.13.

2.42.1 (7 Points)

Take the weighted average: 0.7*2 + 0.1*6 + 0.2*3 = 2.6

2.42.2 (7 Points)

For a 25% improvement, we must reduce the CPI to 2.6*0.75 = 1.95. Thus, we want 0.7*x + 0.1*6 + 0.2*3 < = 1.95. Solving for x shows that the arithmetic instructions must have a CPI of at most 1.07.