East West University

Semester: Spring-2025

Course Title: Computer Architecture

Course Code: CSE360 **Sec: 03**

Assignment on Chapter 1, 2, 3

Submitted by-

Sheikh Sarafat Hossain

2022-3-60-109

Submitted to-

Md. Ezharul Islam, PhD

Professor

Department of Computer Science & Engineering

Date of Submission: 9th April 2025

1.1: Ans:

Four other type of computer:

(1) Personal computers:
* General purpose, variety of software
* Subject to cost/performance tradeoff

(2) Server computers:
* Network based
* High capacity, performance, reliability

(3) SuperComputer:
* High-end scientific and engineering calculations
* Highest capability but represent a small fraction of the overall computer market.

(4) Embedded computers:
* Hidden as components of system
* Stringent power/performance/cost

1.2 : Ans :

a] Assembly lines in automobile manufacturing

→ Performance via pipelining.

b] Suspension bridge cables

→ Dependability via redundancy.

c] Aircraft and marine navigation systems that incorporate wind information.

→ Performance via prediction.

d] Express elevators in buildings

→ Make the common case fast.

e] Library reserve desk

→ Hierarchy of memories.

f] Increasing the gate on a CMOS transistor to decrease its switching time.

→ Performance via parallelism.

g] Adding electromagnetic aircraft catapults (which are electrically-powered as opposed to current steam-powered models), allowed by the increased power generation offered by the new reactor technology.

→ Design for Moore's Law.

h] Building self-driving cars whose controls systems partially rely on existing sensor systems already installed into the base vehicle, such as lane departure systems and smart cruise control systems.

→ Use Abstraction to simplify design.

1.3 : Ans :

Step 1 : High level language such as C into assembly language.

Step 2 : Assembly language converts into machine language.

1.4 : Ans :

a] The minimum size of frame = frame(Pixels) * bytes

= 1280 * 1024 * 3

= 3932160

b] Time = $\frac{\text{size}}{\text{speed}}$

here,

size = 3932160 x 8

= 31457280 bit

speed = 10^8 bit/s

Time = $\frac{31457280}{10^8}$

= 0.3145 s

1.5 : Ans :

a] We know,

$$\text{Performance} = \frac{\text{clock rate}}{\text{CPI}}$$

$\therefore P_1$ Performance $= \frac{3 \times 10^9}{1.5} = 2 \times 10^9$

$\therefore P_2$ " $= \frac{2.5 \times 10^9}{1} = 2.5 \times 10^9$

$\therefore P_3$ " $= \frac{4 \times 10^9}{2.2} = 1.8 \times 10^9$

thus P_2 processor has the highest performance

b] We know,

CPU cycle = execution time * clock rate

$\therefore P_1$ cycle $= 10 \times 3 \times 10^9 = 3 \times 10^{10}$

$\therefore P_2$ " $= 10 \times 2.5 \times 10^9 = 2.5 \times 10^{10}$

$\therefore P_3$ " $= 10 \times 4 \times 10^9 = 4 \times 10^{10}$

Again,

number of instruction = $\frac{\text{cpu cycle}}{\text{CPI}}$

∴ No. of instructions for $P_1 = \frac{3 \times 10^{10}}{1.5} = 2 \times 10^{10}$

∴ No. of " " $P_2 = \frac{2.5 \times 10^{10}}{1} = 2.5 \times 10^{10}$

∴ No. of " " $P_3 = \frac{4 \times 10^{10}}{2.2} = 1.8 \times 10^{10}$

c] time = 10, then reducing time by 30%

$= \frac{10 \times 30}{100} = 3s$

∴ After reducing time = 10 − 3 = 7s

Now,

increasing CPI by 20%

New CPI for $P_1 = \frac{1.5 \times 120}{100} = 1.8$

" " " $P_2 = \frac{1 \times 120}{100} = 1.2$

" " " $P_3 = \frac{2.2 \times 120}{100} = 2.6$

Now,

$$\text{New clock rate} = \frac{\text{cycle} \times \text{New CPI}}{\text{time}}$$

$\therefore P_1$ Clock rate $= \frac{2\times10^{10}\times1.8}{7} = 5.14\ \text{GHz}$

$\therefore P_2$ " " $= \frac{2.5\times10^{10}\times1.2}{2} = 4.28\ \text{GHz}$

$\therefore P_3$ " " $= \frac{1.8\times10^{10}\times2.6}{2} = 6.75\ \text{GHz}$

1.6 : Ans:

here,

P_1 total cpu time $= (10^5 + 2\times10^5\times2 + 5\times10^5\times3 + 2\times10^5\times3)/(2.5\times10^9)$

$= 10.4\times10^{-4}\ \text{S}.$

P_2 cpu time $= (10^5\times2 + 2\times10^5\times2 + 5\times10^5\times2 + 2\times10^5\times2)/(3\times10^9)$

$= 6.66\times10^{-4}\ \text{S}$

We know,

Global CPI = (CPU time x clock rate) / Ic

P_1 CPI $= 10.4 \times 10^{-4} \times 2.5 \times 10^9 / 10^6 = 2.6$

P_2 CPI $= 6.66 \times 10^{-4} \times 3 \times 10^9 / 10^6 = 2.0$

b]

We know,

cpu clock cycle $= (\Sigma(Ic \times CPI))$

$\therefore P_1$ clock cycle $= (10^5 \times 1) + (2 \times 10^5 \times 2) + (5 \times 10^5 \times 3) + (2 \times 10^5 \times 3) = 26 \times 10^5$

$\therefore P_2$ clock cycle $= (10^5 \times 2) + (2 \times 10^5 \times 2) + (5 \times 10^5 \times 2) + (2 \times 10^5 \times 2)$

$= 20 \times 10^5$

1.7 : Ans:

a)

We know,

$$CPI = \frac{\text{cpu time}}{\text{instructions} \times \text{cycle time}}$$

$$\therefore A\ CPI = \frac{1.1}{1 \times 10^9 \times 10^{-9}} = 1.1$$

$$\therefore B\ CPI = \frac{1.5}{1.2 \times 10^9 \times 10^{-9}} = 1.25$$

b)

We know,

$$\text{clock rate ratio} = \frac{\text{Execution A}}{\text{Execution B}}$$

$$= \frac{1.1}{1.5}$$

$$= 0.733$$

$$\therefore \text{clock Rate Difference} = \frac{1}{\text{clock rate ratio}}$$

$$= \frac{1}{0.733}$$

$$= 1.36.$$

c) Compare with A,

$$\frac{\text{Number of instruction A}}{\text{Number of instruction (new)}} = \frac{1 \times 10^{-9}}{6 \times 10^{-8}}$$

$$= 1.67$$

Compare with B,

$$\frac{\text{Number of instruction B}}{\text{Number of instruction (new)}} = \frac{1.2 \times 10^{-9}}{6 \times 10^{-8}}$$

$$= 2.22$$

1.10.1 : Ans :

We know,

$$\text{die area}_{15} = \frac{\text{Wafer area}}{\text{dies per wafer}} = \frac{\pi \times (7.5)^2}{64}$$

$$= 2.1 \text{ cm}^2$$

$$\text{Yield}_{15} = \frac{1}{\left(1 + \left(\text{Defect per area} \times \frac{\text{Die area}}{2}\right)\right)^2}$$

$$= \frac{1}{\left(1 + \left(0.02 \times \frac{2.1}{2}\right)\right)^2}$$

$$= 0.9593$$

Again,

$$\text{die area}_{20} = \frac{\pi \times (10)^2}{100} = 3.14 \text{ cm}^2$$

$$\text{Yield}_{20} = \frac{1}{\left(1 + \left(0.03 \times \frac{3.14}{2}\right)\right)^2} = 0.9093$$

1.10.2 : Ans :

We know,

$$\text{cost per die} = \frac{\text{Cost per Wafer}}{(\text{dies per wafer} \times \text{yield})}$$

$$\therefore \text{cost per dies}_{15} = \frac{12}{84 \times 0.9593} = 0.1469$$

$$\therefore \text{cost per dies}_{20} = \frac{15}{100 \times 0.9593} = 0.1650$$

1.10.3 : Ans :

Dies per wafer is increased by 10%

Defect " " " " 15%

$$\therefore \text{die area}_{15} = \frac{\pi \times (7.5)^2}{92.4} = 1.911 \text{ cm}^2$$

$$\therefore \text{yield}_{15} = \frac{1}{\left(1 + \left(0.023 \times \frac{1.91}{2}\right)\right)^2} = 0.9574$$

Again,

$$\text{die area}_{20} = \frac{\pi \times (10)^2}{110} = 2.856 \text{ cm}^2$$

$$\text{yield}_{20} = \frac{1}{\left(1 + \left(0.034 \times \frac{2.85}{2}\right)\right)^2} = 0.9095$$

1.11.1: Ans:

$$\text{CPI} = \frac{1}{0.333} \times 750 \times (2389 \times 10^9)$$

$$= 0.94$$

1.11.2: Ans:

$$\text{Spec ratio} = \frac{\text{ref time}}{\text{Exec time}} = \frac{9650}{750} = 12.86$$

1.11.5: Ans:

$$\text{Spec ratio} = \frac{1}{1.156} = 0.86$$

1.11.6 : Ans :

$$CPI = \frac{\text{CPU time} \times \text{clock rate}}{\text{No. of instruction}} = \frac{200 \times 4 \times 10^{-9}}{86 \times 2.389 \times 10^{12}}$$

$$= 1.37$$

1.11.9 : Ans :

$$\text{No. of instruction} = \frac{\text{cpu time} \times \text{clock rate}}{CPI}$$

$$= \frac{260 \times 0.9 \times 4 \times 10^{9}}{1.61}$$

$$= 2.14 \times 10^{12}$$

1.12.1 : Ans :

$$P_1 = \frac{5 \times 10^{9} \times 0.9}{4 \times 10^{9}} = 1.125$$

$$P_2 = \frac{1 \times 10^{9} \times 0.9}{3 \times 10^{9}} = 0.3$$

∴ clock rate of P_1 > clock rate of P_2.

1.12.3 : Ans :

We know,

$$MIPS = \frac{\text{clock rate} \times 10^{-6}}{CPI}$$

$$\therefore P_1 \quad MIPS = \frac{4 \times 10^9 \times 10^{-6}}{0.9} = 4.4 \times 10^3$$

$$= 4.4 \times 10^3$$

$$\therefore P_2 \quad MIPS = \frac{3 \times 10^9 \times 10^{-6}}{0.75} = 4 \times 10^3$$

$$P_1 \quad MIPS > P_2 \quad MIPS$$

2.1:

Ans:

```
addi $t0, $S1, -5
add $S0, $1, $t0
```

2.2:

Ans:

```
f = i+(g+h);
```

2.3:

Ans: B[8] = A[i-j]

```
sub $t0, $S3, $S4
sll $t0, $t0, 2
add $t0, $S6, $t0
lw $t1, 0($t0)
sw $S1, 32($S7)
```

2.4.

Ans:

1. sll $t0, $s0, 2 # $t0 = f * 4
2. add $t0, $s6, $t0 # $t0 = &A[f]
3. sll $t1, $s1, 2 # $t1 = g * 4
4. add $t1, $s7, $t1 # $t1 = &B[g]
5. lw $s0, 0($t0) # f = A[f]
6. addi $t2, $t0, 4 # $t2 = &A[f+1]
7. lw $t0, 0($t2) # $t0 = A[f+1]
8. add $t0, $t0, $s0 # $t0 = A[f+1] + A[f]
9. sw $t0, 0($t1) # B[g] = A[f+1] + A[f]

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

2.5:

Ans:

In the above MIPS code we can only remove the 6th line and modify line 7 by,

7. lw $t0, 4($t0)
8. add $t0, $t0, $s0
9. sw $t0, 0($t1)

2.7:

Ans:

Value → 0xabcdef12

Little Endian

Address	Data
3	ab
2	cd
1	ef
0	12

Big Endian

Address	Data
3	12
2	ef
1	cd
0	ab

2.8:

Ans:

$0xabcdef12_{(16)}$ in decimal $\rightarrow 2882400018_{(10)}$

2.9:

Ans:

B[8] = A[i] + A[j]

```
sll $t0, $s3, 2   # $t0 = i * 4
add $t0, $t0, $s6 # $t0 = &A[i]
lw  $t0, 0($t0)   # $t0 = A[i]
sll $t1, $s4, 2   # $t1 = j * 4
add $t1, $t1, $s6 # $t1 = &A[j]
lw  $t1, 0($t1)   # $t1 = A[j]
add $t0, $t0, $t1 # t0 = A[i] + A[j]
sw  $t0, 32($s7)  # B[8] = A[i] + A[j]
```

2.10:

Ans:

```
addi $t0, $S6, 4    # $t0 = A[1]
add $t1, $S6, $0    # $t1 = A[0]
sw $t1, 0($t0)      # $t0 = A[0]
lw $t0, 0($t0)      # $t0 = A[0]
add $S0, $t1, $t0   # f = A[0] + A[0]
```

$$f = A[0] + A[0]$$

2.11:

Ans:

Instruction	type	op	rs	rt	rd	immediate
addi $t0, $S6, 4	I-type	8	22	8	—	4
add $t1, $S6, $0	R type	0	22	0	9	—
sw $t1, 0($t0)	I-type	43	8	9	—	0
lw $t0, 0($t0)	I-type	35	8	8	—	0
add $S0, $t1, $t0	R-type	0	9	8	16	—

2.12.1:

Ans:

$S_0 \rightarrow 0x80000000$

$S_1 \rightarrow 0xD0000000$

∴ add $t_0, $s_0, $s_1

value of $t_0 \rightarrow 0x150000000$

2.12.2:

Ans:

The result in t_0 is overflow.

2.12.3:

Ans:

sub $t_0, $s_0, $s_1

$0xB0000000$

2.12.4:

Ans: No overflow

2.12.5

Ans:

0x1D000000

2.12.6:

Ans: overflow

2.13.1:

Ans:

add $t0, $s0, $s1

The memory allocated to one instruction is 4 bytes.

The range of number can be calculated as 2^{n-1} $n=32$

The Range is -2^{32-1} to 2^{32-1}

∴ Range of $s1 = -2147483648 to 2147483648

∴ Range of $1 = 2147483647 - 128

= 2147483519

2.13.2:

Ans:

sub $t0, $S0, $S1

Range of $S1 = 128 - 2147483647

= -2147483519

2.13.3:

Ans:

sub $t0, $S0, $S1

range of $S1 = -2147483648 + 128

= -2147483520

2.16:

sub $t3, $S3, $V0

op=0, rs=3, rt=2, rd=3, shamt=0, funct=34

000000	00011	00010	00011	00000	100010
0	3	2	3	0	34

op=6 bit, rs=5 bit, rt=5 bit, rd=5 bit, shamt=5 bit, funct=6 bit

2.17:

lw $V0, 4($V1)

op = 0x23, rs = 1, rt = 2, const = 0x4

10001	00001	00010	0000000000000100
0x23	1	2	0x4

2.18.1:

Ans:

In R-type instructions, opcode would be 8 bits, rs, rt, rd fields would be 2 bits each.

2.18.2:

Ans:

In the I-type instructions, opcode would be 8 bits, rs, rt would be 2 bits each.

2.19.1 :

Ans:

sll $t2, $t0, 44

The instruction performs shift left logical which shifts the content of $t0 left by 44 bits and store it in $t2.

$t2 = 0xAAAAAAAA B

or $t2, $t2, $t1

Now, the value of $t2 and $t1 perform the logical operation OR and stone it in $t2.

AAAAAAAA OR 12345678

$t2 = 0xBABEFEF8.

2.19.2:

Ans:

srl $t_2, $t_0, 4

The instruction srl will make logical right shift on $t_0, then the value of $t_2 will be,

$t_2 = 0xAAAAAAA0

and $t_2, $t_2, -1

the value of $t_2 will perform the logical operations AND with immediat -1

$t_2 = AAAAAAA0.

2.19.3

Ans:

srl $t2, $t0, 3

The instruction srl will make logical right shift on $t0, the value of $t2 will be,

$t2 = 0x15555555

and $t2, $t2, 0xFFEF

Now, the value of $t2 will be perform the logical AND with 0xFFEF;

0x15555555 AND 0xFFEF

$t2 = 00005545

2.20:

Ans:

```
srl  $t0, $t0, 11
srl  $t0, $t0, 26
ori  $t2, $0, 0x03ff
sll  $t2, $t2, 16
ori  $t2, $t2, 0xffff
and  $t1, $t1, $t2
or   $t1, $t1, $t0
```

2.21:

Ans:

```
nor  $t1, $t2, $t2
```

2.22:

Ans:

```
lw  $t3, 0($s1)
sll $t1, $t3, 4
```

2.23:

Ans:

$t2 = 3

2.24:

Ans:

```
Jump: no, beg:    no
```

2.25.1:

Ans: i-type.

2.25.2:

Ans:

```
addi $t2, $t2, -1
beq $t2, $0, loop
```

2.26.1:

Ans: 20.

2.26.2:

```
i = 20;
do{
    B += 2;
    i = i-1;
} while (i>0);
```

2.26.3:

Ans: 5*N

2.27: Ans:

```
        addi $t0, $0, 0
        beq $0, $0, TEST1

Loop1:  addi $t1, $0, 0
        beq $0, $0, TEST2

Loop2:  add $t3, $t0, $t1
        sll $t2, $t1, 4
        add $t2, $t2, $s2
        sw $t3, ($t2)
        addi $t1, $t1, 1

TEST2:  slt $t2, $t1, $s1
        bne $t2, $0, Loop2
        addi $t0, $t0, 1

TEST1:  slt $t2, $t0, $s0
        bne $t2, $0, Loop1
```

2.28: Ans:

14 instructions to be implement
and 158 instructions executed.

2.29:

```
for (i=0; i<100; i++) {
    result = MemArray[so];
    so = so + 4;
}
```

2.30:

Ans:

```
    addi $t1, $s0, 400
Loop: lw $s1, 0($t1)
      add $s2, $s2, $s1
      addi $t1, $t1, -4
      bne $t1, $s0, Loop
```

2.32:

Ans:

Due to the recursive nature of the code, it is not possible for the compiler to in line the function call.

2.35:

Ans:

We can use tail-call optimization for the second call to function. But then we must restore $ra, $s0, $s1 and $sp before that call. we save only one instructions (jr, $ra).

2.38:

Ans:

0x00000011

2.39: Ans:

Generally all solution are similar:

lui $t1, top 16-bits

ori $t1, $t1, botton 16-bits.

2.40:

Ans:

No jump can go upto 0x0FFFFFFC

2.41:

Ans:

No, range is;

0x604 + 0x1FFFC = 0x00020600

to 0x604 - 0x20000 = 0xFFFE-0604.

2.42:

Yes. range is 0x1FFFF004 + 0x1FFFC

= 0x2001F000 to 0x1FFFF004

- 0x20000 = 1FFDF004.

2.45:

Ans:

It is possible for one or both processors to complete the code without ever reaching the sc instructions. If only one executes sc, it completes same cycle; but one sc completes first and then the other detects this and fails.

2.46.2:

Ans:

107.04%, 1113.43%

2.47.1:

Ans: 2.6

2.47.2:

Ans:

0.88

2.47.3:

Ans: 0.533333333...

3.1: Ans:

$5ED4_{(16)} - 07A4_{(16)}$

$$\begin{array}{r} 5ED4 \\ -07A4 \\ \hline 5730 \end{array}$$

3.2: Ans:

The given values are,

$(5ED4)_{16} = (0101\ 1110\ 1101\ 0100)_2$

$(07A4)_{16} = (0000\ 0111\ 1010\ 0100)_2$

for signed numbers is sign magnitute format, those two number are positive. That's why $5ED4 - 07A4 = 5730$.

So, the result in signed 16-bit hexa-decimal numbers in sign-maghitute format is 5730.

3.3: Ans:

The given value is 5EDA where represent the hexadecimal $(5EDA)_{16}$

$= (0101\ 1110\ 1101\ 1010)_2$

The attraction of hexadecimal is that the numbering system contains 16 different character (0-9, A-F)

It allows representing large binary numbers in more compactly where each hexadecimal represents 4bit binary number and make it easier for reader.

3.4: Ans:

$(4365)_8 \rightarrow$ 5000 1111 0101

$(3412)_8 \rightarrow$ 0111 0000 1010

$(753)_8 \rightarrow$ 0001 1110 1011

So, in unsign 12 bit octal numbers is $(753)_8$

3.5: Ans:

Maghitude of 4365 $\rightarrow (365)_8$ or $(245)_{10}$

Magnitude of 3412 $\rightarrow (412)_8$ or $(266)_{10}$

To subtract 3412 from 4365 in sigh magnitude format, $(365)_8 - (412)_8 = (-377)_8$

3.6: Ans:

$(185)_{10} - (122)_{10} = (63)_{10}$

Indicate neither.

3.7: Ans:

The given number in binary,

$(185)_{10} = 1011\ 1001$, $(122)_{10} = 0111\ 1010$

Assume that, these numbers are sign then we get, $(-57)_{10} = (1010\ 1001)_2$ from 185, ab the MSB indicates the sign bit.

Now,

$(-57)_{10} + (122)_{10} = 65$ (Neither)

3.8: Ans:

$(-57)_{10} = (122)_{10} = (-179)_{10}$

This is not fit with 8 bit sign magnitude format.

3.9: Ans:

$(151)_{10} = 1001\ 0111$

$(214)_{10} = 1101\ 0110$

from these two value; we get (-105) and (-42).

$(-105) + (-42) = (-147)_{10}$. As the range is -128.

3.10: Ans:

here,

$(-105) - (-42) = -63$.

3.11: Ans:

$151 + 214 = 365$, where 8 bit unsigned range is 255.

3.12: Ans:

Given that 6 bit unsigned integer 62 and 12 multiplication:

Iteration	Steps	Multiplier	Multiplicand	Product
0	Initial	001 010	000 000 110 010	000 000 000 000
1	lsb = 0, noop Lshift Mcand Rshift Mer	000 101	000 000 100 100	000 000 000 000
2	lsb = 1 Prod = Prod + Mcand lshift Mcand Rshift Mer	000 010	000 011 001 000	000 001 100 100
3	lsb = 0, noop lshift Mcand Rshift Mper	000 001	000 110 010 000	000 001 100 100
4	lsb = 1 Prod = Prod + Mcand lshift mcand Rshift Mper	000 000	011 100 100 000	000 111 110 100
5	lsb = 0, op	000 000	011 011 000 000	000 111 110 100
6	lsb = 0, noop	000 000	110 010 000 000	000 111 110 100

So the product is $(000\ 111\ 110\ 100)_2 = (764)_{16}$

3.13: Ans: Hexadecimal multiplication of 62 x 12,

Iteration	Step	Multiplier	Multiplicand	Product
0	Initial	0110 0010	0000 0000 0001 0010	0000 0000 0000 0000
1	lsb=0, no op lshift Mcand Rshift Mplier	0011 0001	0000 0010 0010 0100	0000 0000 0000 0000
2	lsb=1; Prod = Prod + Mcand lshift Mcand Rshift Mplier	0001 1000	0000 0000 0100 1000	0000 0000 0000 0000
3	lsb=0 lshift Mcand Rshift Mplier	0000 1100	0000 0000 1001 0000	0000 0000 0010 0100
4	lsb=0 lshift Mcand Rshift Mplier	0000 0110	0000 0001 0010 0000	0000 0000 0010 0100
5	lsb=0 lshift Mcand Rshift Mplier	0000 0011	0000 0010 0100 0000	0000 0000 0010 0100
6	lsb=1 Prod = Prod + Mcand lshift Mcand Rshift Mplier	0000 0001	0000 0100 1000 0000	0000 0010 0110 0100
7	lsb=1 Prod = Prod + Mcand lshift Mcand Rshift Mplier	0000 0000	0000 0100 1000 0000	0000 0110 1110 0100
8	lsb=0	0000 0000	0001 0010 0000 0000	0000 0110 1110 0100

So, the product is $(0000\ 0110\ 1110\ 0100)_2$

$= (6E4)_{16}$.

3.14: Ans:

For hardware loop takes $(3 \times A)$ cycles.

for software, loop takes $(5 \times A)$ cycles,

So, $(3 \times 8) \times 4tu = 96$ times units, for hardware

and $(5 \times 8) \times 4tu = 160$ times units, for software.

3.15: Ans:

It takes B times units to get through an adder and there will be A-1 adder where word is 8 bits requiring 7 adders $(8-1)$

$7 \times 4tu = 28$ times.

3.15: Ans:

It takes B time units to get through an adder and the adders are arranged in a tree structure as it works for faster multiplication.

It required $\log_2(A)$ levels which is 3 levels so, 8 bit wide word required 7 adders in 3 levels $(3 \times 4tu) = 12tu$.

3.16: Ans:

Iteration	Steps	Quotient	Divisor	Remainder
0	Initial	000 000	001 001 000 000	000 000 111 100
1	Rem = Rem - Div Rem < 0; R+D, Q << Rshift Div	000 000	001 000 100 000	000 000 111 100
2	Rem = Rem - Div Rem < 0, R+D Q << Rshift	000 000	000 100 010 000	000 000 111 100
3	Rem = Rem - Div Rem < 0, R+D Q << Rshift Div	000 000	000 010 001 000	000 000 111 100
4	Rem = Rem - Div Rem < 0, R+D Q << Rshift Div	000 000	000 001 000 100	000 000 111 100
5	Rem = Rem - D Rem < 0, R+D Q << Rshift div	000 000	000 000 100 010	000 000 111 100
6	Rem = Rem - D Rem > 0, Q << 1 Rshift Div	000 001	000 000 010 001	000 000 011 010
7	Rem = Rem - Div Rem > 0, Q << 1 Rshift Div	000 0011	000 000 001 000	000 000 001 001

3.19: Ans:

Iteration	Steps	Divisor	Remainder/Quotient
0	Initial	010 001	000 000 111 100
1	R<< Rem = Rem-Div Rem<0, R+D	010 001	000 001 111 000
2	R<< Rem = Rem-D R<0, R+D	010 001	000 011 110 000
3	R<< Rem = Rem-Div R<0, R+D	010 001	000 111 100 000
4	R<< Rem = Rem-Div R<0, R+D	010 001	001 111 000 000
5	R<< Rem = Rem-Div Rem>0, R0 = 1	010 001	001 101 000 001
6	R<< Rem = Rem-Div Rem≥0, R0 = 1	010 001	001 001 000 011

3.20: Ans:

Given: 0x0C000000 = 0000 1100 0000 0000 0000 0000 0000 0000

the decimal number and the unsigned integer it does represent is 201326592.

3.22: Ans:

Given, 0x0c000000 = 0000 1100 0000 0000 0000 0000 0000

here, the sign bit = 0

the exponent = 24 - 127 = -103

and significant = 0

we know that, $(1.0 + 0) \times 2^{-103} = 1.0 \times 2^{-103}$.

3.23 : Ans.

The binary representation of 6325 is $1111111.01 = 1.1 \times 2^5$

sign positive, exp = 127 + 5 = 132

final pattern = 0 1000 0100 1111 1010 0000 0000 0000 0000

3.24: Ans:

$63.24 \times 10^0 = 111111.01 \times 2^0$

Normalizing; $1.11111\,01 \times 2^5$

sign = 0 and exponent = 1023 + 5

= 1028

Where 1 bit for sign, 11 bits for exponent and 52 bit for significand.

3.25: Ans:

$63.25 \times 10^{10} = 3F.4 \times 16^{0}$

for normalized 0.340×10^{2}

sign = 0, As single precision, exp = 64+2 = 66

Where 1 bit for sign, 8 bits for exponent and 23 bit for significant.

3.26: Ans:

$-1.5625 \times 10^{-1} = -0.15625 \times 10^{0}$

$= -0.00101 \times 2^{0}$

Normalizing, -0.101×2^{-2}

exponent = -2 fraction = -0.101...-0

Here, ex = 2047 + (-2) = 2045

3.27: Ans:

$-1.5625 \times 10^{-1} = -0.15625 \times 10^{0}$

$= 1.01 \times 10^{-3}$

[redacted]

exponent $= -3 + 15 = 12$

3.30: Ans:

Given,

$-8.054825 \times 10^{0} \times (-1.79931640625 \times 10^{-1})$

$= 1.449293137$

$-8.054825 \times 10^{0} = -1.00000001111 \times 2^{1}$

$-1.79931640625 \times 10^{-1} = -1.0111000010 \times 2^{-1}$

Exp: $-3+3=0$; $0+16=16$ (10000)

both has negative sign

$$1.0000000111$$
$$\times 1.0111000010$$
$$1.0111001100000100111 0$$

Guard = 0, Round = 0, Sticky = 1

3.32: Ans:

Given,

$(3.9843725\times10^{1} + 3.4375\times10) + 1.271\times10^{3}$

here, $0.39843725 = 0.0110011000000$

$\therefore A = 1.10011000000\times2^{-2}$

$0.34325 = 0.0101100000000\ldots$

$\therefore B = 1.0110000000\times2^{-2}$

$1.271\times10^{3} = 1271 = 0b\ 1111\ 1011$

$= 1.1011101011\times2^{10}$

A) 1.1001100000

(B) + 1.0111000000

10.1111100000

Normalize, (A+B) = 1.0111110000 × 2^1

(A+B)+C = 0 1 1 0 1 0 1 0 1 1 1 0 0 0 = 1772.

3.33° Ans°

$3.984375 \times 10^{-1} + (3.4375 \times 10^{-1} + 1.271 \times 10^{3})$

A, $(0.3984375) = 1.10011000000 \times 2^{-2}$

B, $(0.34375) = 1.01100000000 \times 10^{-2}$

C, $1.271 \times 10^{3} = 1271 = 1.10111010111 \times 2^{10}$

$(B+C) = +1.10111010111$

$A + (B+C) = +1.10111010111 \times 2^{10}$

$= 01101010111010111 = 1771.$