HW #1

Chapter 1 Concepts in Digital Systems

[1.1]
(a) $2^2 = 4$ (b) $2^5 = 32$ (c) $2^6 = 64$

[1.2]
(a) 16MB = 8×2^{24} bits = 1.34×10^8 bits
(b) 64MB = 8×2^{26} bits = 5.37×10^8 bits
(c) 256MB = 8×2^{28} bits = 2.15×10^9 bits

[1.3]
(a) 8.1GB = $8.1\times8\times2^{30}$ bits = 6.96×10^{10} bits
(b) $(8.1\times8\times2^{30})/32 = 2.17\times10^9$ words

[1.4]
A set of 8 compass directions requires 3 bits to represent in a binary word, *direction*.

Direction	*direction*
N	000
S	001
E	010
W	011
NE	100
NW	101
SE	110
SW	111

Additional 4 directions would require one more bit in a binary code word, *direction*, to represent all the directions.

Direction	*direction*
N	0000
S	0001
E	0010
W	0011
NE	0100
NW	0101
SE	0110
SW	0111
NNE	1000
NNW	1001
SSE	1010
SSW	1011

Note that the binary codewords, 1100 – 1111, are unused.

[1.5]
The minimum size of the binary word, *shade_of_white*, that will cover all 43 different shades is 6 bits which accounts for 64 available codewords. Thus, there will be 21 unused binary codewords which can be reserved for the future use in case that there are more shades added.

[1.6]
(a) W=0101=5
(b) X=1100=12
(c) Z=1001=9

[1.7]
(a) 101010=42
(b) 011011=27
(c) 110001=49
(d) 011011=27

[1.8]
(a) A=01010101=85
(b) B=11001100=204
(c) C=10100011=163

[1.9]

Bit	Weighting	Bit	Weighting
b_0	1	b_8	256
b_1	2	b_9	512
b_2	4	b_{10}	1,024
b_3	8	b_{11}	2,048
b_4	16	b_{12}	4,096
b_5	32	b_{13}	8,192
b_6	64	b_{14}	16,384
b_7	128	b_{15}	32,768

[1.10]
(a) 0101 1000 1110 1010= 0x58EA = 22,762
(b) 0100 1101 0001 0100= 0x4D14 = 19,732
(c) 1000 1000 1000 1000= 0x8888 = 34,952

[1.11]
(a) 0x1F5ACC6A = 526,044,266
(b) 0x465F5A04 = 1,180,654,084

[1.12]
(a) 8=1000
(b) 14=1110
(c) 23=0001 0111
(d) 36=0010 0100
(e) 18=0001 0010
(f) 9=1001
(g) 16=0001 0000

[1.13]
(a) 32=0010 0000
(b) 42=0010 1010
(c) 76=0100 1100
(d) 67=0100 0011
(e) 95=0101 1111

HW #1

[1.14]
(d) 103=0110 0111
(e) 155=1001 1011
(f) 225=1110 0001
(g) 187=1011 1011
(h) 283=0001 0001 1011
(i) 384=0001 1000 0000

[1.15]
(a) 0x1F = 0001 1111=31
(b) 0xA8=1010 1000=168
(c) 0x7B= 0111 1011=123
(d) 0x67= 0110 0111=103

[1.16]
(a) 0x1F20 = 0001 1111 0010 0000=7,968
(b) 0x0ABC = 0000 1010 1011 1100=2,748
(c) 0x70D2 = 0111 0000 1101 0010=28,882
(d) 0x86BA = 1000 0110 1011 1010=34,490

[1.17]
$2^9 < 1000 < 2^{10}$ Thus, the smallest number of bits needed to represent a number that is less than or equal to 1000 is 10 bits.

[1.18]
The maximum number that a 6-digit calculator can represent is 999,999.
$2^{19} = 524,288 < 999,999 < 2^{20} = 1,048,576$
Hence, at least 20-bit binary word size is required.

[1.19]
(a) 0.10010=0.56250
(b) 0.11010=0.81250
(c) 0.01011=0.34375
(d) 0.10101=0.65625

[1.20]
(a) 1101.0110=13.3750
(b) 1001.0011=9.1875
(c) 0101.1001=5.5625

[1.21]
(a) 0.500=0.100
(b) 0.550=0.100011001100110011
(c) 0.625=0.101
(d) 0.650=0.1010011001100110011

[1.22]
(a) The 4-bit binary fraction closest to 0.6 is 0.1010 =0.625
(b) Error = 0.025 = 4.17%
(c) The 8-bit binary fraction closest to 0.6 is 0.10011010 =0.6015625.
Error = 0.0015625 = 0.26%.

[1.23]
A round-off error less than 0.3%= 0.009423
At least 4 bits would be needed to obtain the binary representation with a round-off error less than 0.3%. That binary representation is 0.0101 = 0.3125 with an error of 0.0016.

[1.24] $f = 1/T = 1/(0.3\mu s) = 3.33$ MHz

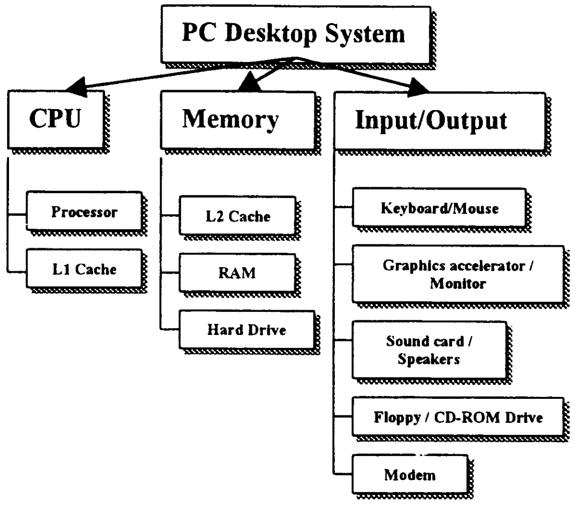
[1.25] $f = 1/T = 1/(150\ ns) = 6.67$ MHz

[1.26] $T = 1/f = 1/(400\ MHz) = 2.5$ ns

[1.27]
$T_{500MHz} = 2$ ns and $T_{550MHz} = 1.82$ ns
The decrease in clock period = 0.18 ns or about 9% decrease.

[1.28]
The following specification can be commonly found in an advertisement for a personal computer in a newspaper or magazine:

- Pentium®II or III-class processor at the clock frequency between 233MHz-500MHz with 32KB L1 cache
- 512KB/1MB L2 cache
- 32MB – 128MB RAM
- 4GB – 16GB hard drive
- 3.5" Floppy drive
- 24X – 40X CD-ROM drive
- 4MB – 8MB AGP graphics accelerator
- 15" – 21" SVGA Monitor
- 56Kbps V.90 Internal modem
- 16-bit sound card
- Speakers
- Keyboard
- 2- or 3-botton Mouse
- Windows® 98 or NT Operating System



HW #1

[8.29]

(a) 14 1110
6 0110
⇒ 2s complement: 1010
1 1 1
1 1 1 0 +
1 0 1 0
1 **1 0 0 0** = **8**

(b) 34 = 0010 0010
21 = 0001 0101
⇒ 2s complement: 1110 1011
1 1 1   1
0 0 1 0 0 0 1 0 +
1 1 1 0 1 0 1 1
1 **0 0 0 0 1 1 0 1** = **13**

(c) 134 = 1000 0110
62 = 0011 1110
⇒ 2s complement: 1100 0010
1   1 1
1 0 0 0 0 1 1 0 +
1 1 0 0 0 0 1 0
1 **0 1 0 0 1 0 0 0** = **72**

(d) 196 = 1100 0100
118 = 0111 0110
⇒ 2s complement: 1000 1010
1
1 1 0 0 0 1 0 0 +
1 0 0 0 1 0 1 0
1 **0 1 0 0 1 1 1 0** = **78**

[8.30]

The following is the logic table for 1-bit addition:

Inputs			**Outputs**	
A_n	B_n	C_n	S_n	C_{n+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

After logic simplication,
$S_n = (A_n \oplus B_n) \oplus C_n$
$C_{n+1} = A_nB_n + (A_n \oplus B_n)C_n$

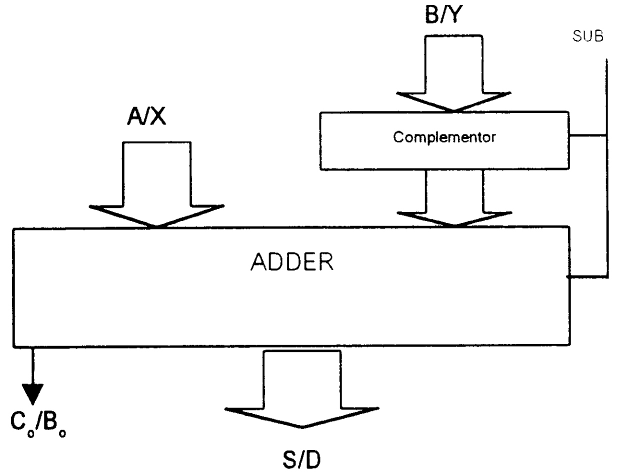
The following is the logic table for 1-bit subtraction:

Inputs			**Outputs**	
X_n	Y_n	B_n	D_n	B_{n+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

After logic simplification,
$D_n = X_n \oplus (Y_n \oplus B_n)$ (Same as S_n in addition)
$B_{n+1} = Y_nB_n + (Y_n \oplus B_n)\overline{X_n}$

From the equations above, it shows the similarity between the two logic required for a 1-bit adder and a 1-bit subtractor.

The correct question to be asked is what is the advantage of using the 2s complement method over the borrowing method in subtraction. The answer is that using the 2s complement method will enable us to use the same hardware for both addition and subtraction with an additional circuitry for complementor which is usually much smaller than adding another substractor as shown by the block diagram below:



[8.31]

Bit	*s*	a_2	a_1	a_0
Decimal weight	-8	4	2	1

Hw#1

[8.32]
(a)

```
        1 0 1 0 (10) ×
        1 0 1 1 (11)
        1 0 1 0
      1 0 1 0
  1 0 1 0
  1 1 0 1 1 1 0 (110)
```

(b)

```
        1 1 1 0 (14) ×
        0 0 1 0 ( 2)
      1 1 1 0
      1 1 1 0 0  (28)
```

(c)

```
        1 0 1 1 (11) ×
        0 1 1 1 ( 7)
        1 0 1 1
      1 0 1 1
    1 0 1 1
  1 0 0 1 1 0 1 (77)
```

(d)

```
        0 1 0 0 ( 4) ×
        1 0 1 1 (11)
        0 1 0 0
      0 1 0 0
  0 1 0 0
  0 1 0 1 1 0 0 (44)
```

[8.33]

```
            0 0 1 1  0 1 0 0 (0x34) ×
            0 0 0 1  0 1 1 1 (0x17)
            0 0 1 1  0 1 0 0
        0   0 1 1 0  1 0 0
      0 0   1 1 0 1  0 0
  0 0 1 1   0 1 0 0
  0 1 0 0   1 0 1 0  1 1 0 0 (0x4AC)
```

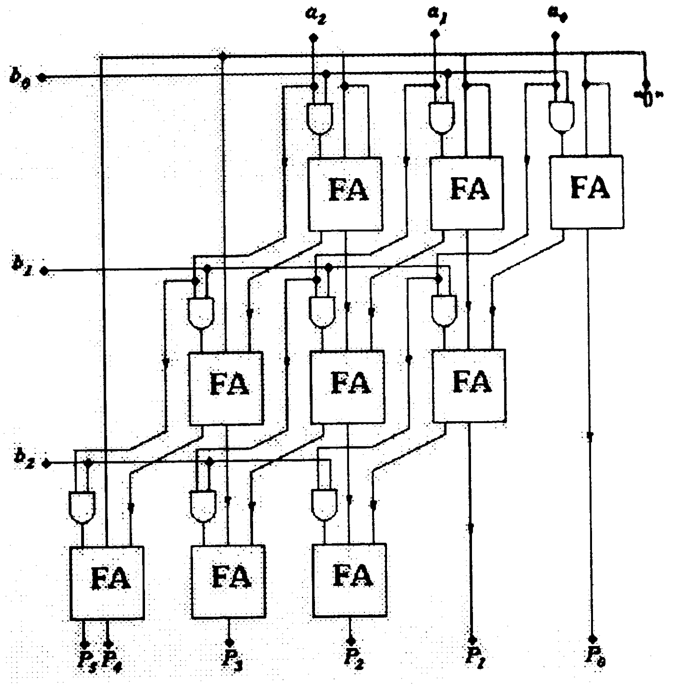
[8.34] (a)

			a_2	a_1	a_0	×
			b_2	b_1	b_0	
c_5	c_4	c_3	c_2	c_1		
			a_2b_0	a_1b_0	a_0b_0	
		a_2b_1	a_1b_1	a_0b_1		
	a_2b_2	a_1b_2	a_0b_2			
p_5	p_4	p_3	p_2	p_1	p_0	

$p_0 = a_0b_0$
$p_1 = a_1b_0 \oplus a_0b_1 \oplus c_1$
$p_2 = a_2b_0 \oplus a_1b_1 \oplus a_0b_2 \oplus c_2$
$p_3 = a_2b_1 \oplus a_1b_2 \oplus c_3$
$p_4 = a_2b_2 \oplus c_4$
$p_5 = c_5$

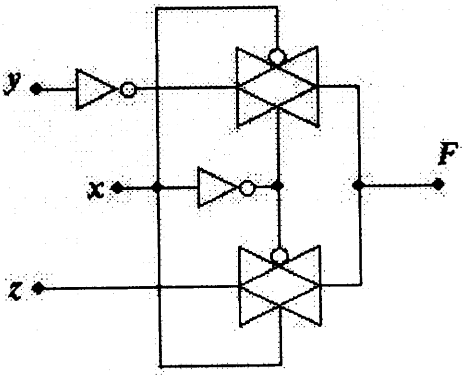
Note that c_i are the carry bits from the previous bit in addition.

(b)



[8.35] Students who understand the concept of the transmission gate network well enough would be able to tell that there is something wrong about the question. Clearly, it would make more sense to design the transmission gate for the following functions:
$F = \overline{x} \cdot \overline{y} + x \cdot z$ or $F = x \cdot \overline{y} + \overline{x} \cdot z$
rather than $F = x \cdot \overline{y} + x \cdot z$

The first network below shows the transmission gate network for $F = \overline{x} \cdot \overline{y} + x \cdot z$:



HW#1

1) 100.01 + 101 11

```
  1 1
 100.01
 101.11
 ------
1010.00
```

2)

a) 10111 X 110

```
    10111
      110
    -----
    00000
   10111
   ------
   10111
  10111
 --------
 10001010
```

b) 101 1 X 11.01 = 10011.111

```
    1011
    1101
    ----
    1011
   0000
   ----
   0101
  1011
  ----
  1101
 1101
 ---------
 100 1 1.111
```

3) HEX → 1796B.ADD

OCT → 274553.5335

4

1) 32 − 113

32 → 00100000 113 → 01110001

−113 → 10001111

$$\begin{array}{r} 00100000 \\ 10001111 \\ \hline 10101111 \end{array}$$

−ve no.

2s complement

01010001 → 64 + 16 + 1 → $\underline{\underline{-81}}$

2) 45 − 127

45 → 00101101 127 → 01111111

−127 → 10000001

$$\begin{array}{r} 00101101 \\ 10000001 \\ \hline 10101110 \end{array}$$

−ve no.

2s complement.

01010010 → 64 + 16 + 2 → $\underline{\underline{-82}}$

HW#1

5.

```
   21  →   00010101          4 → 00000100

  -4   →   11111100

X  21  →        10101
          ------------
             11111100
           00000000
          ------------
          111111110
          11111100
         ------------
          11111011
        00000000
       ------------
         11111101
       11111100
      ------------
      1111110101100
      ↗
  -ve no.
```

2s complement

01010100 → 64 + 16 + 4 = -84