210 Data Representation

Some important dates
1936: Alan Turing –
    The Turing machine, computability, universal machine
1946 – Eckert and Archly
    ENIAC – first electronic computer (with vacuum tubes)
1947 John von Neumann
    First to describe modern day computer, with central processor, memory, output, input
•Early days of electronics, analog computers provided a relatively inexpensive means for solving complex problems (1950's and 60's)
In an analog computer, a voltage (or current) is used to represent a data value. Variables are then operated on by linear circuit elements to compute an output function. A voltage waveform may represent a time varying function, with the operations of integration and differentiation being relatively easily achieved. The basic building block of an analog computer was a linear amplifier. As with any hi-fi amplifier, the better the linearity, the lower the noise levels, and the smaller the distortion the better the accuracy of the computer. Even with state of the art electronics analog computers would rarely achieve 0.1% accuracy!
•Today's computers are built from transistors
- Transistor is either off or on → Need to represent numbers using only off and on
- Two symbols off and on can represent the digits 0 and 1

BIT is Binary Digit
→We need to know how to represent binaries, transform, operate, etc…

Number representation (Decimal)

Digits (or symbols) allowed: 0-9
Base (or radix): 10
The order of the digits is important as 345 is:
3 x 100 + 4 x 10 + 5 x 1
3 x 10**2 + 4 x 10**1 + 5 x 10**0
- 3 is the most significant symbol (it carries the most weight)
- 5 is the least significant symbol (it carries the least weight)

In a more general way:
$137.06 = 1 \times 10^{**}2 + 3 \times 10^{*}1 + 7 \times 10^{*}0 + 0 \times 10^{-1} + 6 \times 10^{-2}$
- The powers of ten are determined by the position relative to the decimal point.

Using positional coefficients and weights we can express any weighted number system in the following generalized form:
$X = x_n w_n + x_{n-1} w_{n-1} + \ldots + x_{-1} w_{-1} + \ldots + x_{-m} w_{-m}$
where
- $w_i = r^i$ and $0 \le x_i \le (r - 1)$
- The r^i are the weighted values.
- r is the radix or base.
- The x_i are the positional coefficients.

Number representation (Binary-1)

Digits (symbols) allowed: 0, 1

Base (radix): 2 each binary digit is called a BIT

The order of the digits is significant

Numbering of the digits:

From MSB (bit n-1) to LSB (Bit 0) where n is the number of digits in the number

MSB stands for most significant bit

LSB stands for least significant bit

Number representation (Binary-2)

1001 (base 2) is:

1001_2= 1 x 2**3 + 0 x 2**2 + 0 x 2**1 + 1 x 2**0

9 (base 10) or 9_{10}

11001 (base 2) is:

11001_2 = 1 x 2**4 + 1 x 2**3 + 0 x 2**2 + 0 x 2**1 + 1 x 2**0

25 (base 10) or 25_{10}

For the alpha machine (and for simulator as well) no native function to represent binaries (why ??)

Represent 110011_2 in base 10 → 51_{10}

Represent 51_{10} in binary

My method: subtract largest power of 2 smaller than 51 until you reach 1

- largest power of 2 smaller than 51: 32 -> 51-32=19
- largest power of 2 smaller than 19: 16 -> 19-16=3
- largest power of 2 smaller than 3: 2 -> 3-2=1
- 51_{10}=32+16+2+1=2**5+2**4+2*1+2*0= 110011_2

Number representation (Octal)

Digits (symbols) allowed: 0, 1,…7

Base (radix): 8

The order of the digits is important

345 (base 8) is:

$345_8 = 3 \times 8^{**}2 + 4 \times 8^{**}1 + 5 \times 8^{**}0 = 192 + 32 + 5 = 229_{10}$

1001 (base 8) is:

$1001_8 = 1 \times 8^{**}3 + 0 \times 8^{**}2 + 0 \times 8^{**}1 + 1 \times 8^{**}0\ 512 + 0 + 0 + 1$

513 (base 10)

Easy: Transform from Binary to Octal

$229_{10}=128+64+32+4+1=11100101_2$ (011 100 101)->345_8

→ Best way to transform from decimal to octal is to go via Binary

Octal representation to binary representation

$345_8 = 3 \times 8^{**}2 + 4 \times 8^{**}1 + 5 \times 8^{**}0$

In Binary: 011 100 101 → 011100101_2

Number representation (hexadecimal)

Digits (symbols) allowed: 0-9, a-f

Base (radix): 16

The order of the digits is important

hex	decimal	binary
0	0	0000
1	1	0001
. . .		
9	9	1001
a	10	1010
b	11	1011
c	12	1100
d	13	1101
e	14	1110
f	15	1111

$a3_{16}$ or 0xa3

(1010 0011) 10100011_2

(010 100 011)--> 243_8

a3 (base 16) is:

$a3_{16}$ (0xa3) = a x 16**1 + 3 x 16**0 = 160 + 3 = 163 (base 10)

Transformations between hexadecimal-binary-octal-decimal use binary as intermediate data representation

Bits/Bytes and Words

- A bit is a single value, either 1 or 0.
- A byte is an ordered sequence of 8 bits.
  - memory is typically *byte addressed* – each byte has a unique address.
- A word is an ordered sequence of bits, the length depends on the processor architecture.
  - typical value for modern computers is 32 (64 for the alpha) bits.

Byte Values

- There are 256 different byte values

```
00000000
00000001
00000010
00000011
   …
11111110
11111111
```

Bytes as unsigned integers

- Base 2 number using positional notation

0	0	1	0	0	0	1	1
128	64	32	16	8	4	2	1

most significant bit

Value is $32 + 2 + 1 = 33_{10}$

least significant bit

Hexadecimal

- Values are often expressed in base 16.
- One sequence of 4 bits is reduced to a single hexadecimal *digit*:

binary	0000	0001	0010	0011	0100	0101	0110	0111
hex	0	1	2	3	4	5	6	7

binary	1000	1001	1010	1011	1100	1101	1110	1111
hex	8	9	A	B	C	D	E	F

- A byte is represented by a 2 digits hexadecimal

Basic Data Types

- Integral
  - Stored & operated on in general registers

Intel	**GAS**	**Bytes**	**C**
byte	`b`	**1**	`[unsigned] char`
word	`w`	**2**	`[unsigned] short`
long word	`l`	**4**	`[unsigned] int`
quadword		8	

Hexadecimal and Octal

•Hexadecimal is well suited for byte (8-bit word) oriented machines which are most often organized as 16 or 32 bit words. A word is then 2 or 4 (8-bit) bytes. In 'Hex' a byte is represented by 2 hex digits.
• In the past there have been 12-bit and 36-bit machines (note, the IBM 1620 which used a 6-bit alphanumeric representation).
• Octal offers a natural way to represent 6-bit groups (two octal digits).
• However Octal representation of 16-bit words has the effect of disguising the 8-bit bytes; Example
$A = 101_8 = 01\ 000\ 001_2$
$B = 102_8 = 01\ 000\ 010_2$
$\text{word(AB)} = 01\ 000\ 001\ 01\ 000\ 010_2$
$= 0\ 100\ 000\ 101\ 000\ 010_2$
$= 040502_8$
• Since hexadecimal does not have this problem, it is the favored representation for current machines.
$A = 41_{16}$
$B = 42_{16}$
$AB = 4142_{16}$
and $BA = 4241_{16}$

Transformation (1)

1. **Any base --> decimal**

Use the definition (summation) given above.

134 (base 5) 1 x 5**2 + 3 x 5**1 + 4 x 5**0 25 + 15 + 4 44 (base 10)

2. **Decimal --> ANY base**

Divide decimal value by the base until the quotient is 0.

The remainders give the digits of the value.

Examples:

36 (base 10) to base 2 (binary) → 36 (base 10) == 100100 (base 2)

36/2 = 18 **r=0** <-- lsb 18/2 = 9 **r=0** 9/2 = 4 **r=1** 4/2 = 2 **r=0** 2/2 = 1 **r=0** 1/2 = 0 **r=1** <-- msb

14 (base 10) to base 2 (binary) → 14 (base 10) == 1110 (base 2)

14/2 = 7 **r=0** <-- lsb 7/2 = 3 **r=1** 3/2 = 1 **r=1** 1/2 = 0 **r=1** <-- msb

38 (base 10) to base 3 → 38 (base 10) == 1102 (base 3)

38/3 = 12 **r=2** <-- ls digit 12/3 = 4 **r=0** 4/3 = 1 **r=1** 1/3 = 0 **r=1** <-- ms digit

100 (base 10) to base 5 → 100 (base 10) = 400 (base 5)

100/5 = 20 **r=0** 20/5 = 4 **r=0** 4/5 = 0 **r=4**

Transformation (2)

3. **Binary --> octal**

1. group into 3's starting at least significant symbol (if the number of bits is not evenly divisible by 3, then add 0's at the most significant end)
2. write 1 octal digit for each group:
   - 100 010 111 (binary) 4 2 7 (octal)
   - 10 101 110 (binary) 2 5 6 (octal)

4. **Binary --> hex** (same as binary to octal)

1. group into 4's starting at least significant symbol (if the number of bits is not evenly divisible by 4, then add 0's at the most significant end)
2. write 1 hex digit for each group:
   - 1001 1110 0111 0000 (binary) → 0x9e70
   - 1 1111 1010 0011 (binary) → 0x1fa3

5. **Hex --> binary**

write down the 4 bit binary code for each hexadecimal digit:

- 0x39c8 (hex) → 0011 1001 1100 1000

6. **Octal --> binary**

write down the 8 bit binary code for each octal digit:

- 501_8 (Octal) → 101 000 001

Transformation (3)

7. hex --> octal (2 steps)
    1. hex --> binary
    2. binary --> octal

Where the bases are growing powers of a common value, the transformation in between is straightforward

- binary → base 4, octal → hexadecimal, etc…

Examples:

Transformation from base 3 to base 9

2100122 (base 3)

One base 9 digit is substituted for each 2 base 3 digits.

base 3	9
00	0
01	1
02	2
10	3
11	4
12	5
20	6
21	7
22	8

2 10 01 22 (base 3) → 2318 (base 9)

Exercises

Convert the following decimal numbers first to binary and then to hexadecimal: [4 marks] 27_{10} 121_{10}

Convert the following unsigned hexadecimal numbers to octal: [4 marks] 225516 $11EC3_{16}$

The number 233_{10} is equal to the following:

1. 10101001_2
2. 11001001_2
3. 11111001_2
4. 11101001_2

The number 2338 is equal to the following:

1. 15010
2. 16410
3. 15510
4. 15910