

Data Representation

Ref: 2.1

CompOrg - Data Representation

1

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 bits.
 - word size reflects the size of addresses (a word is typically large enough to hold any address).

CompOrg - Data Representation

2

Byte Values

- There are 256 different byte values

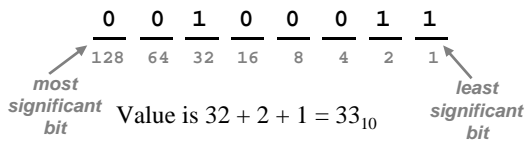
```
00000000
00000001
00000010
00000011
...
11111110
11111111
```

CompOrg - Data Representation

3

Bytes as unsigned integers

- Base 2 number using positional notation



CompOrg - Data Representation

4

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

CompOrg - Data Representation

5

Specifying hex values in C/C++

- **0x07** specifies the 8 bit value **0000 0111**
- **0xF2** specifies the 8 bit value **1111 0010**
- The compiler determines the length (number of bits) depending on the context
 - and assumes leading 0s

CompOrg - Data Representation

6

C Programming and bytes

- Data type **char** is one byte.
 - there are 256 different char values!
- Some operations treat a **char** as an 8 bit integer:

```
char x = 12;  
    x++;  
if (x>3) y++;
```

CompOrg - Data Representation

7

C char data type

- C allows you to specify a byte value using a *character literal*.

```
char x = 'E';  
    x = x+'A';  
    x = x+3;
```

- ASCII codes are used by C to represent text characters, each text character has a unique 8 bit code.

CompOrg - Data Representation

8

Other C integral data types

- Actual length (number of bits) depends on the compiler and machine architecture.
- Common lengths are shown:

short int, unsigned short

– 16 bits (2 bytes)

int, unsigned int

– 32 bits (4 bytes)

long int, unsigned long

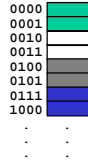
– 32 or 64 bits (4 or 8 bytes)

CompOrg - Data Representation

9

16 bit integers and memory

- A 16 bit (2 byte) word is stored in adjacent memory locations.
 - typically the smaller address must be divisible by 2.
- There are 2 possible arrangements of the bytes:
 - most significant is at lower address
 - most significant is at higher address



Byte Ordering

- Not all processor architectures use the same byte ordering!
- Big-Endian: The most significant byte has the lowest address
- Little Endian: The least significant byte has the lowest address
- Byte ordering is an issue with all multi-byte integers (32 bit and 64 bit).

Programming and byte order

- Generally a programmer doesn't need to know which byte ordering is used!
 - network programs are one exception.
 - debugging assembly language programs often requires knowledge of byte order.

C Programming and data

- It's easy to write C programs that can print out some data in various formats.
- Check out the sample code on the course home page .
- You need to understand *casting*, *pointers* and the C *printf* function.

printf

- C doesn't have cout (or cin, or any object-oriented stuff that is in C++!).

```
int printf(char *fmt, ...);
```

printf examples

```
char x = 35;
```

```
printf("x is %d\n", x);
```

print x as decimal value

```
printf("x is %2x\n", x);
```

print x as hex value

```
printf("x is %c\n", x);
```

print x as ASCII character

Strings in C

- C does not have a *string* data type!
- There are some functions that treat a sequence of bytes (**char**) as a string.
 - each character is specified as the ASCII encoded value found in one byte.
- A string is terminated with the byte value 0.

CompOrg - Data Representation

19

Strings and C

- C functions that deal with strings generally only *know* the starting address of the string.
 - step through memory one byte at a time until a byte with value 0 is found.
- Pointers!

CompOrg - Data Representation

20

Sample Code

```
char *s = "Pointers are fun!";  
printf("The length of the string is %d\n",  
       strlen(s) );  
  
printf("The string is: %s\n", s);
```

CompOrg - Data Representation

21
