S1 Introduction to Computer Systems

1.1 Systems Programming

Virtual memory – large amount of contiguous memory (as it appears to the process)

Physical memory – fragmented memory stored in main memory and on disk

1.2 Basics of Programming

Primitive data types: int, float, double, char

S2 Data Representation

2.1 Primitive Data Types

Bit – 0 or 1

Byte – 8 bits

Nibble – 4 bits

1 nibble maps to 1 hexadecimal digit

Unsigned char = 8 bits, 1 byte. Range $0\ to\ 255$.

Unsigned int = 32 bits, 4 bytes. Range $0\ to\ (2^{32}-1)$.

Unsigned short int = 16 bits, 2 bytes. Range $0\ to\ (2^{16}-1)$.

Signed char = 8 bits, 1 byte. Range $(-128)\ to\ 127$.

Signed int = 32 bits, 4 bytes. Range $\left(\frac{-2^{32}}{2}\right)\ to\ \left(\frac{2^{32}}{2}-1\right)$.

Signed short int = 16 bits, 2 bytes. Range $\left(\frac{-2^{16}}{2}\right)\ to\ \left(\frac{2^{16}}{2}-1\right)$.

Float = 32 bits, 4 bytes. Can represent up to 8 digits.

- 1 sign bit
- 8 bits for e
- 23 bits for f

Double = 64 bits, 8 bytes.

- 1 sign bit
- 11 bits for e
- 52 bits for f

**Always add 127 to e

ASCII – each character is contained in one byte (8 bits).

Bitwise operators

- ~ NOT
- & AND
- | OR
- ^ XOR
- >> right shift
- << left shift

Bitmask operations

- Set Nth bit → `a = a | (1<<N)`
- Clear Nth bit → `a = a & (~(1 << N))`
- Read Nth bit → `return (a & (1 << N) >> N`

Non-decimal prefixes

- Hexadecimal – 0x
- Octal – 0
- Binary – 0b

2.2 Compound Data Types

Strings must be NULL terminated – `‘\0’`

2.3 Pointers

Pointers occupy 4 bytes

Asterisk – variable declaration (indicates data type is a pointer), dereferencing operator. Returns value at the address pointed to by the pointer.

Memory map – table listing all variables: names, values and addresses.

Each memory address represents 1 byte – the first byte in compound data types/dynamically allocated memory.

Arrow operator – equivalent to dereferencing pointer then using dot operator.

S3 Memory Management

3.1 Stack and Heap

OS allocates 4 areas of memory on startup:

Code segment – program instructions, addresses of functions
Data segment – global variables, static variables, literals
Function call stack – manages order of function calls, stores local variables
Heap – part of the data segment, stores all dynamically allocated memory

3.2 Dynamic Memory Allocation

Returns pointer to start address of block of reserved memory (void type).

`malloc(size)` – size in bytes

`calloc(num_items, item_size)` – sets allocated memory to 0.

`free(ptr) –` pointer must be to the beginning of a block of dynamically allocated memory.

Double pointers – allows changing pointer values in called function.