Trends in Computer Architecture

Observed that the number of transistors on a chip double every 18 months.
Memory capacity doubles every 2 years.
Disk capacity doubles every year.
Components: CPu, Memory, System Bus, IO bus, storage, peripherals.
Moore's law: number of transistors on a chip double every 18 months.
Von Neumann Model: single memory space store both instructions and data.
Multicore crisis: diminishing returns for multiple core clock speed.
Power Wall: Power consumption increases with clock speed. it levels out because cooling and heat become an issue **Memory Wall:** Memory access time is slower than CPU speed.

C info

Pointer: int *p; (declare pointer which points to an int), p = &x; (point to x), *p = 5; (set x to 5). &x is the address of x.
Array: int a[20]; (declare array of size 20), int *ptr = a; (pointer to array) acts as a[0].
Multi Dimensional Array: int a[2][3]; (2D array), int *ptr = a; (pointer to 2D array) acts as a[0]. a[0] ∼ first row, a[0][0] ∼ first element of first row. *(m+2)+1 = &m[2][1]
Memory: Malloc: create objects on the heap: int *numbers = (int *)malloc(sizeof(int) * n); free to deallocate. free(numbers)
Struct: struct point { int x; int y; }; struct point p; p.x = 5; p.y = 10; if we pass a pointer of a struct we do struct point *p; p = (struct point *)malloc(sizeof(struct point)); p→x = 5; p→y = 10;
Strings: Treat as array of chars followed by null. ”\0”. `strcpy`(s, ”bar”) to copy value but not same object. `strlen`(s) to get length. `strcmp`(s, t) to compare. `strcat`(s, t) to concatenate.
Input/Output: `scanf` and `printf` for input and output. %d for int, %f for float, %s for string. add f to the front of most operation to put them to a file.

Assembly

Cycle: Fetch, Decode, Execute. CISC ∼ complex, RISC ∼ simple. `opcode src dest` where opcode is the operation, src is the source, dest is the destination.
Registers: EAX, EBX, 32 bit, AX, BX, 16 bit, AH, AL, 8 bit. end x is like variable, end p/i is pointer/index.
Type of operands `movl $eax, %ebx` copy contnet of %eax to %ebx; `movl $0x1, %eax` copy 0x1 to %eax; `movl %eax, 0x1` copy %eax to memory location 0x1; `movl (%ebp, %esi), %eax` copy value at address = ebp + esi to %eax; `movl 8(%ebp, %esi), %eax` copy value at address = ebp + esi + 8 to %eax; `movl 0x80 (%ebx, %esi, 4), %eax` copy value at address = ebx + esi*4 + 0x80 to %eax
Operations: `movl, src, dest` dest = src. `pushl src` esp = esp-4 then move M[esp] = src `popl src` src = M[esp] then esp = esp+4. `leal` compute address using addressing mode without accessing memory **Flags** `ZF` zero flag it is zero, `SF` sign flag if it is negative, `OF` overflow flag if it is too big in 2s complement, `CF` carry flag if it is too big in unsigned.

Number theory

Hexadecimal: $A_{16} = 10_{10}$, $F_{16} = 15_{10}$, $10_{16} = 16_{10}$, Uses 16 bits so 2 bytes. (think float = 32 bit, 4 bytes, 4 digits)
Binary: $1010_2 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 = 10_{10}$
Sign-Magnitutde: First bit is sign, rest is magnitude. $-5_{10} = 1001_2$ Arithmatic is odd, distinct ±0
1's Complement: Negate by flipping bits. $-5_{10} = 1110_2$ Arithmatic is mostly normal, distinct ±0
2's Complement: Negate by flipping bits and adding 1. $-5_{10} = 1011_2$ Arithmatic is normal, distinct 0
IEEE: sign, exp, mantissa. → sign $\cdot\ 2^{e-b} \cdot$ mantissa Bias: e = num exp bits, b = $2^{e-1} - 1$. subnormal, with all 0 exp leads to 0 as first digit
Special values: 0 = all zeros, $\pm\infty$ = all ones in exp, 0 in mantissa, change sign. NaN = all ones in exp, non-zero mantissa.