# Bits and Bytes

CENG331 - Computer Organization

Midterm date
December 2nd
6 pm

**Instructor:**

Murat Manguoglu (Section 1)

Unless otherwise noted adapted from slides of the textbook: http://csapp.cs.cmu.edu/

# Today: Bits and Bytes

- **Compiling, linking and executing:  Hello  World**
- Representing information as bits
- Bit-level manipulations

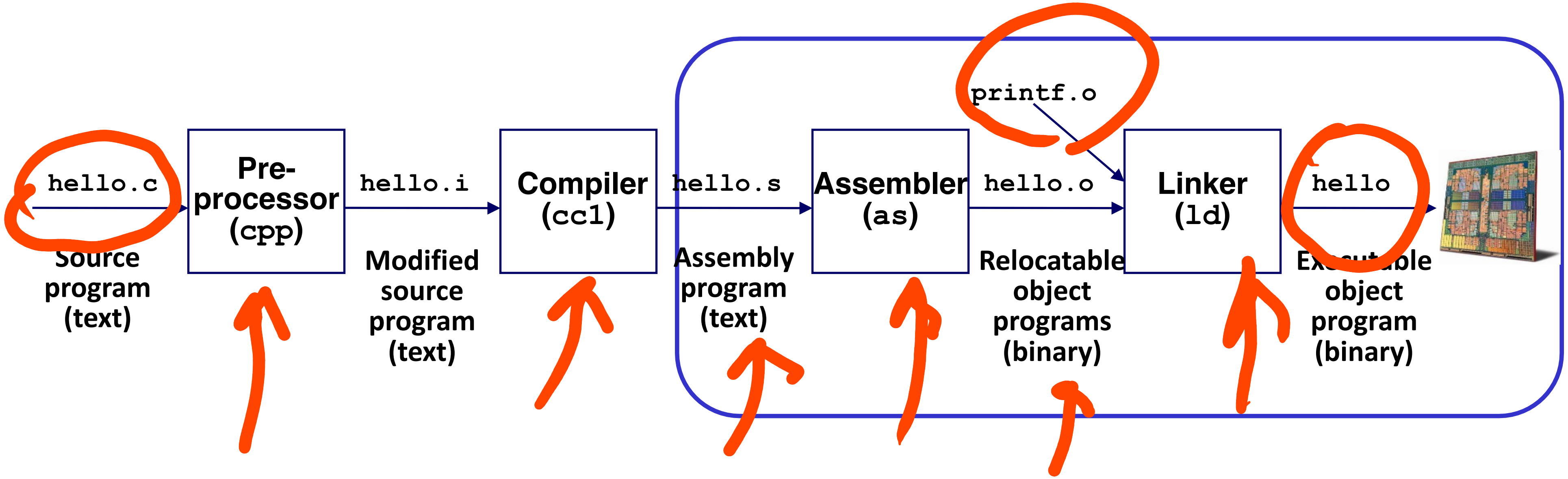
# Hello World!

- What happens under the hood?

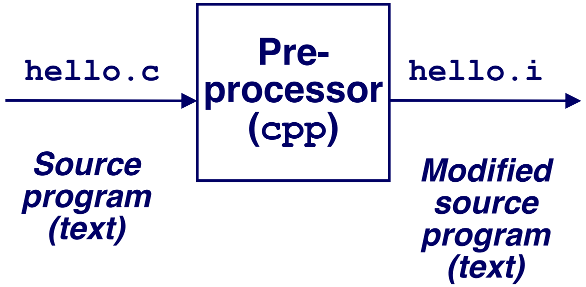
# hello.c

```
#include <stdio.h>

int main() {
  printf(“Hello World”);
}
```

# Compilation of hello.c



# Preprocessing



hello.c

Pre-processor (cpp)

hello.i

*Source program (text)*

*Modified source program (text)*

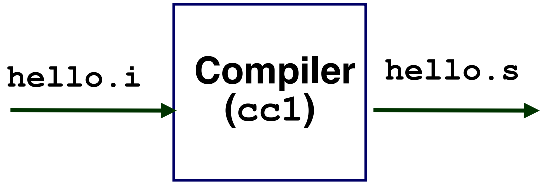
```
#include <stdio.h>


int main() {

    printf(“Hello
    World”);

}
```

cpp hello.c > hello.i

```
# 1 "hello.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 31 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3
4
# 32 "<command-line>" 2
# 1 "hello.c"
# 1 "/usr/include/stdio.h" 1 3 4
...
typedef unsigned char __u_char;
typedef unsigned short int __u_short;
typedef unsigned int __u_int;
typedef unsigned long int __u_long;
...
int main() {
 printf("Hello World");
}
```

# Compiler

```
# 1 "hello.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 31 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 32 "<command-line>" 2
# 1 "hello.c"
# 1 "/usr/include/stdio.h" 1 3 4
...
typedef unsigned char __u_char;
typedef unsigned short int __u_short;
typedef unsigned int __u_int;
typedef unsigned long int __u_long;
...
int main() {
 printf("Hello World");
}
```

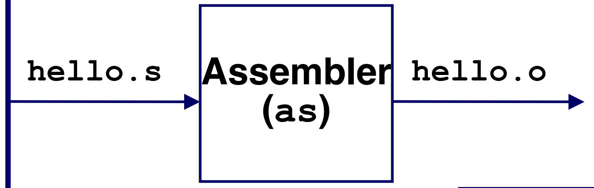


gcc -Wall -S hello.i > hello.s

```
 .file   "hello.c"
        .text
        .section        .rodata
.LC0:
        .string "Hello World"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    .LC0(%rip), %rdi
        movl    $0, %eax
        call    printf@PLT
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .ident  "GCC: (Debian 8.3.0-6)
8.3.0"
        .section        .note.GNU-
stack,"",@progbits
```

# Assembler

```
 .file    "hello.c"
        .text
        .section        .rodata
.LC0:
        .string "Hello World"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    .LC0(%rip), %rdi
        movl    $0, %eax
        call    printf@PLT
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .ident  "GCC: (Debian
8.3.0-6) 8.3.0"
        .section
.note.GNU-stack,"",@progbits
```

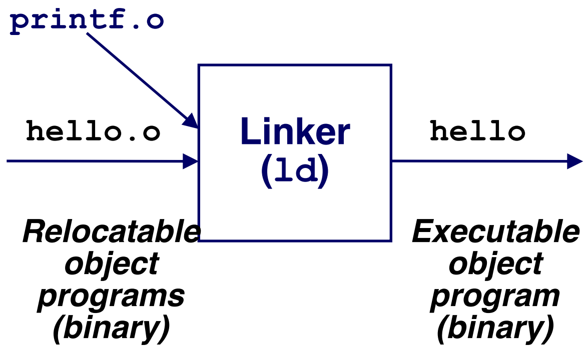


as hello.s -o hello.o

```
0000000 del   E   L   F stx soh soh nul nul nul nul nul nul nul nul nul
0000020 soh nul   > nul soh nul nul nul nul nul nul nul nul nul nul nul
0000040 nul nul nul nul nul nul nul nul   @ stx nul nul nul nul nul nul
0000060 nul nul nul nul   @ nul nul nul nul nul   @ nul  cr nul  ff nul
0000100   U   H  ht   e   H  cr   = nul nul nul nul   8 nul nul nul nul
0000120   h nul nul nul nul   8 nul nul nul nul   ]   C   H   e   l   l
0000140   o  sp   W   o   r   l   d nul nul   G   C   C   :  sp   (   D
0000160   e   b   i   a   n  sp   8   .   3   .   0   -   6   )  sp   8
0000200   .   3   .   0 nul nul nul nul dc4 nul nul nul nul nul nul nul
0000220 soh   z   R nul soh   x dle soh esc  ff bel  bs dle soh nul nul
0000240  fs nul nul nul  fs nul nul nul nul nul nul nul  fs nul nul nul
0000260 nul   A  so dle ack stx   C  cr ack   W  ff bel  bs nul nul nul
0000300 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
0000320 nul nul nul nul nul nul nul nul soh nul nul nul eot nul   q del
0000340 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
0000360 nul nul nul nul etx nul soh nul nul nul nul nul nul nul nul nul
0000400 nul nul nul nul nul nul nul nul nul nul nul nul etx nul etx nul
0000420 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
0000440 nul nul nul nul etx nul eot nul nul nul nul nul nul nul nul nul
0000460 nul nul nul nul nul nul nul nul nul nul nul nul etx nul enq nul
0000500 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
0000520 nul nul nul nul etx nul bel nul nul nul nul nul nul nul nul nul
0000540 nul nul nul nul nul nul nul nul nul nul nul nul etx nul  bs nul
0000560 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
0000600 nul nul nul nul etx nul ack nul nul nul nul nul nul nul nul nul
0000620 nul nul nul nul nul nul nul nul  ht nul nul nul dc2 nul soh nul
0000640 nul nul nul nul nul nul nul nul  fs nul nul nul nul nul nul nul
0000660  so nul nul nul dle nul nul nul nul nul nul nul nul nul nul nul
0000700 nul nul nul nul nul nul nul nul   $ nul nul nul dle nul nul nul
0000720 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
0000740 nul   h   e   l   l   o   .   c nul   m   a   i   n nul   _   G
0000760   L   O   B   A   L   _   O   F   F   S   E   T   _   T   A   B
0001000   L   E   _ nul   p   r   i   n   t   f nul nul nul nul nul nul
0001020 bel nul nul nul nul nul nul nul stx nul nul nul enq nul nul nul
0001040   | del del del del del del del dc1 nul nul nul nul nul nul nul
0001060 eot nul nul nul  vt nul nul nul   | del del del del del del del
0001100  sp nul nul nul nul nul nul nul stx nul nul nul stx nul nul nul
0001120 nul nul nul nul nul nul nul nul nul   .   s   y   m   t   a   b
0001140 nul   .   s   t   r   t   a   b nul   .   s   h   s   t   r   t
0001160   a   b nul   .   r   e   l   a   .   t   e   x   t nul   .   d
0001200   a   t   a nul   .   b   s   s nul   .   r   o   d   a   t   a
0001220 nul   .   c   o   m   m   e   n   t nul   .   n   o   t   e   .
0001240   G   N   U   -   s   t   a   c   k nul   .   r   e   l   a   .
0001260   e   h   _   f   r   a   m   e nul nul nul nul nul nul nul nul
```

od –a hello.o

# Linker


printf.o
hello.o
Linker
(ld)
hello
Relocatable object programs (binary)
Executable object program (binary)

```
0000000 del   E   L   F stx soh soh nul nul nul nul nul nul nul nul nul
0000020 soh nul   > nul soh nul nul nul nul nul nul nul nul nul nul nul
0000040 nul nul nul nul nul nul nul nul   @ stx nul nul nul nul nul nul
0000060 nul nul nul nul   @ nul nul nul nul nul   @ nul  cr nul  ff nul
0000100   U   H  ht   e   H  cr   = nul nul nul nul   8 nul nul nul nul
0000120   h nul nul nul nul   8 nul nul nul nul   ]   C   H   e   l   l
0000140   o  sp   W   o   r   l   d nul nul   G   C   C   :  sp   (   D
0000160   e   b   i   a   n  sp   8   .   3   .   0   -   6   )  sp   8
0000200   .   3   .   0 nul nul nul nul dc4 nul nul nul nul nul nul nul
0000220 soh   z   R nul soh   x dle soh esc  ff bel  bs dle soh nul nul
0000240  fs nul nul nul  fs nul nul nul nul nul nul nul  fs nul nul nul
0000260 nul   A  so dle ack stx   C  cr ack   W  ff bel  bs nul nul nul
0000300 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
0000320 nul nul nul nul nul nul nul nul soh nul nul nul eot nul   q del
0000340 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
0000360 nul nul nul nul etx nul soh nul nul nul nul nul nul nul nul nul
0000400 nul nul nul nul nul nul nul nul nul nul nul nul etx nul etx nul
0000420 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
0000440 nul nul nul nul etx nul eot nul nul nul nul nul nul nul nul nul
0000460 nul nul nul nul nul nul nul nul nul nul nul nul etx nul enq nul
0000500 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
0000520 nul nul nul nul etx nul bel nul nul nul nul nul nul nul nul nul
0000540 nul nul nul nul nul nul nul nul nul nul nul nul etx nul  bs nul
0000560 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
0000600 nul nul nul nul etx nul ack nul nul nul nul nul nul nul nul nul
0000620 nul nul nul nul nul nul nul nul  ht nul nul nul dc2 nul soh nul
0000640 nul nul nul nul nul nul nul nul  fs nul nul nul nul nul nul nul
0000660  so nul nul nul dle nul nul nul nul nul nul nul nul nul nul nul
0000700 nul nul nul nul nul nul nul nul   $ nul nul nul dle nul nul nul
0000720 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
0000740 nul   h   e   l   l   o   .   c nul   m   a   i   n nul   _   G
0000760   L   O   B   A   L   _   O   F   F   S   E   T   _   T   A   B
0001000   L   E   _ nul   p   r   i   n   t   f nul nul nul nul nul nul
0001020 bel nul nul nul nul nul nul nul stx nul nul nul enq nul nul nul
0001040   | del del del del del del del dc1 nul nul nul nul nul nul nul
0001060 eot nul nul nul  vt nul nul nul   | del del del del del del del
0001100  sp nul nul nul nul nul nul nul stx nul nul nul stx nul nul nul
0001120 nul nul nul nul nul nul nul nul nul   .   s   y   m   t   a   b
0001140 nul   .   s   t   r   t   a   b nul   .   s   h   s   t   r   t
0001160   a   b nul   .   r   e   l   a   .   t   e   x   t nul   .   d
0001200   a   t   a nul   .   b   s   s nul   .   r   o   d   a   t   a
0001220 nul   .   c   o   m   m   e   n   t nul   .   n   o   t   e   .
0001240   G   N   U   -   s   t   a   c   k nul   .   r   e   l   a   .
0001260   e   h   _   f   r   a   m   e nul nul nul nul nul nul nul nul
…
```

**gcc hello.o –o hello**

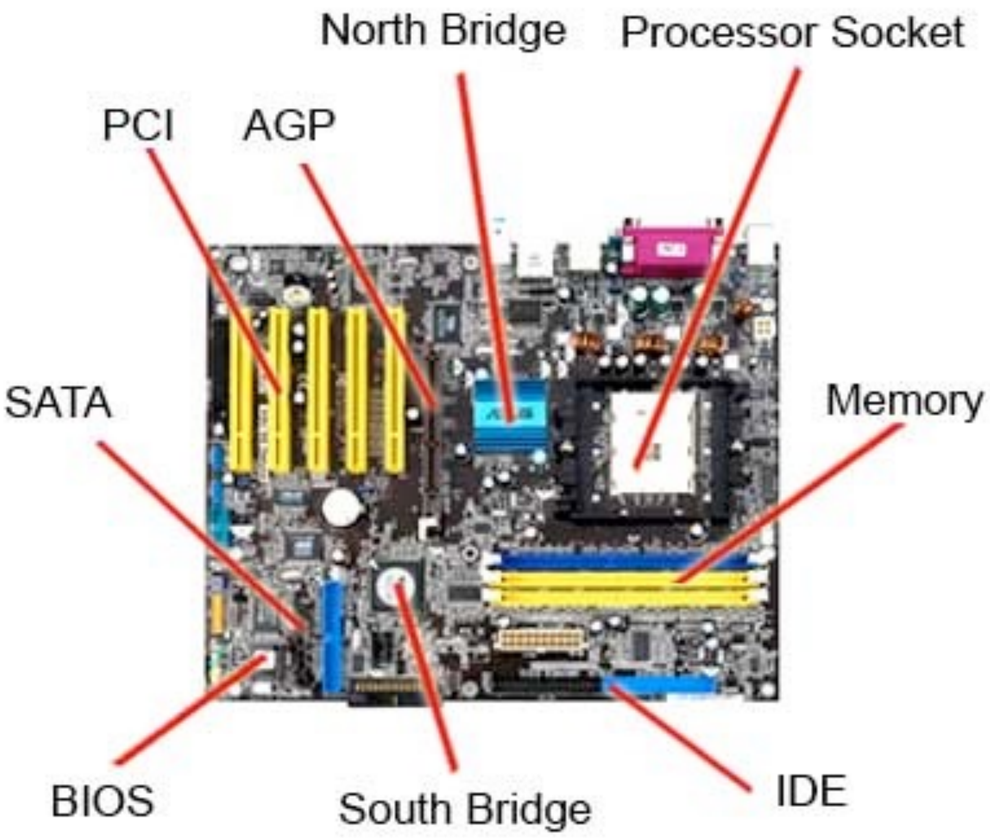
```
0000000 del   E   L   F stx soh soh nul nul nul nul nul nul nul nul nul
0000020 etx nul   > nul soh nul nul nul   P dle nul nul nul nul nul nul
0000040   @ nul nul nul nul nul nul nul   `   9 nul nul nul nul nul nul
0000060 nul nul nul nul   @ nul   8 nul  vt nul   @ nul  rs nul  gs nul
0000100 ack nul nul nul eot nul nul nul   @ nul nul nul nul nul nul nul
0000120   @ nul nul nul nul nul nul nul   @ nul nul nul nul nul nul nul
0000140   h stx nul nul nul nul nul nul   h stx nul nul nul nul nul nul
0000160  bs nul nul nul nul nul nul nul etx nul nul nul eot nul nul nul
0000200   ( stx nul nul nul nul nul nul   ( stx nul nul nul nul nul nul
0000220   ( stx nul nul nul nul nul nul  fs nul nul nul nul nul nul nul
0000240  fs nul nul nul nul nul nul nul soh nul nul nul nul nul nul nul
0000260 soh nul nul nul eot nul nul nul nul nul nul nul nul nul nul nul
0000300 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
0000320   h enq nul nul nul nul nul nul   h enq nul nul nul nul nul nul
0000340 nul dle nul nul nul nul nul nul soh nul nul nul enq nul nul nul
0000360 nul dle nul nul nul nul nul nul nul dle nul nul nul nul nul nul
0000400 nul dle nul nul nul nul nul nul   M soh nul nul nul nul nul nul
0000420   M soh nul nul nul nul nul nul nul dle nul nul nul nul nul nul
0000440 soh nul nul nul eot nul nul nul nul  sp nul nul nul nul nul nul
0000460 nul  sp nul nul nul nul nul nul nul  sp nul nul nul nul nul nul
0000500   X soh nul nul nul nul nul nul   X soh nul nul nul nul nul nul
…
```

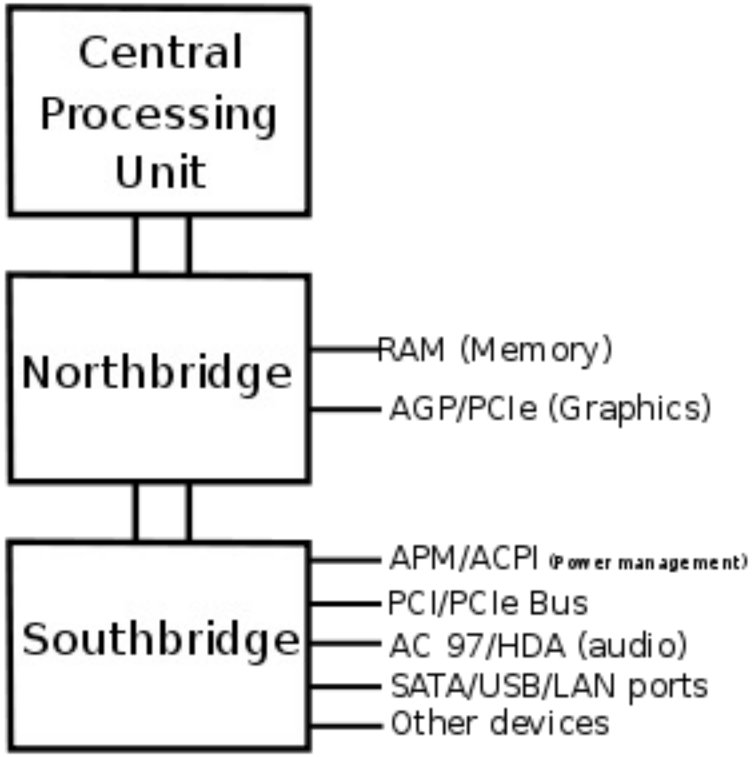
**od –a hello**

# Finally…

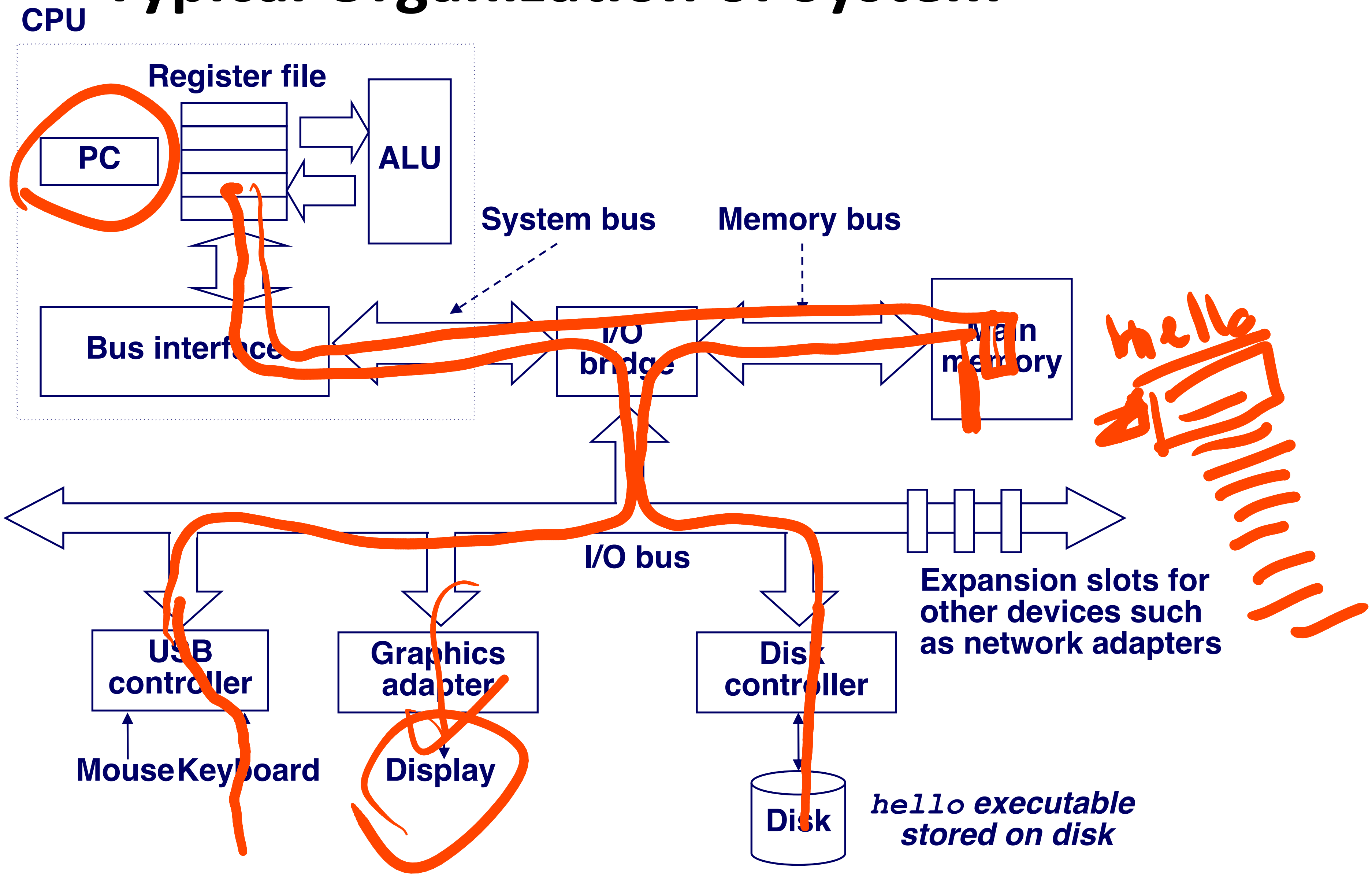
```
$ gcc hello.o -o hello

$ ./hello

Hello World$
```

# How do you say "Hello World"?

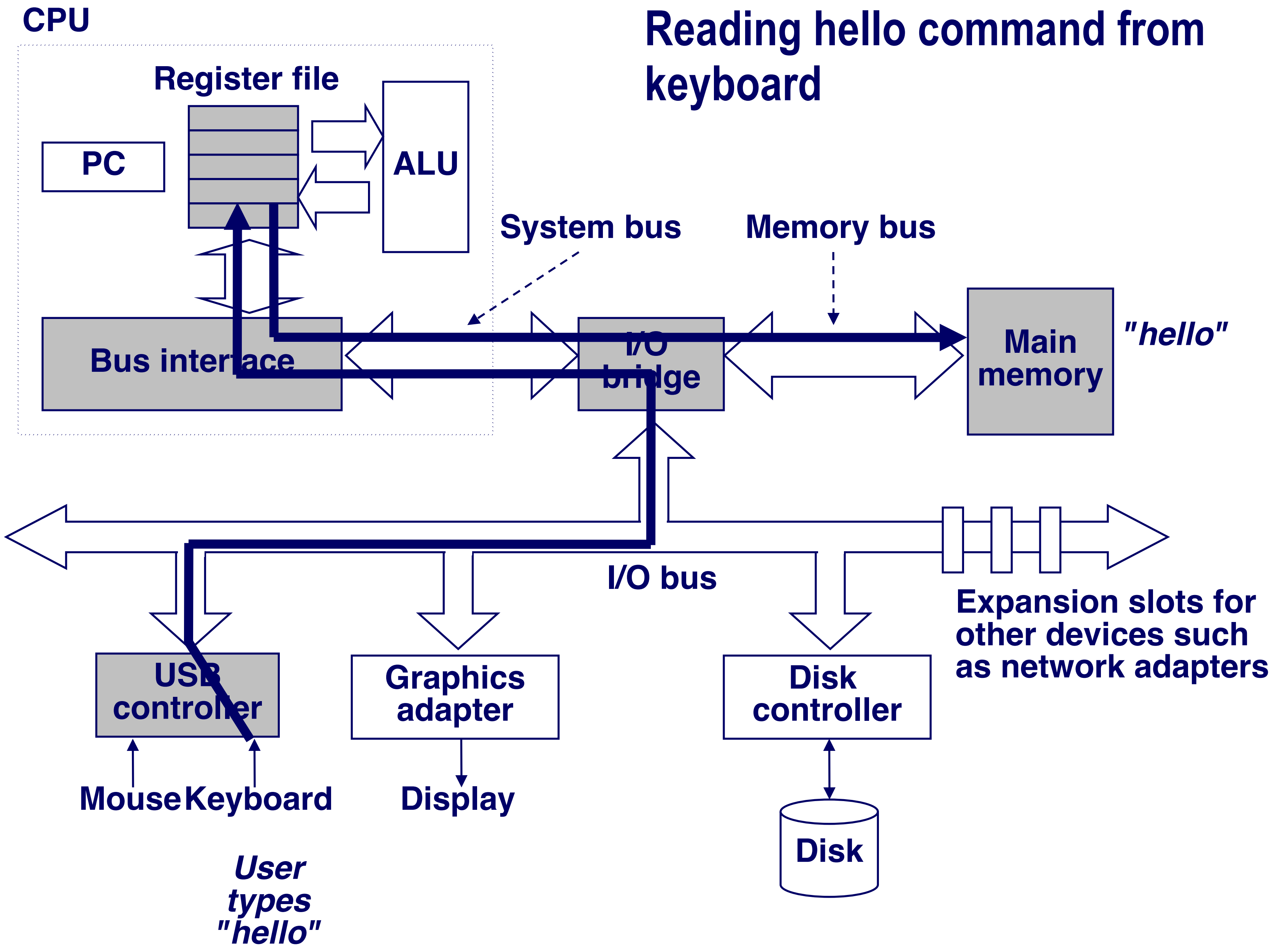




# Typical Organization of System



# Reading hello command from keyboard



# Today: Bits and Bytes

- Compiling, linking and executing:  Hello  World
- **Representing information as bits**
- Bit-level manipulations

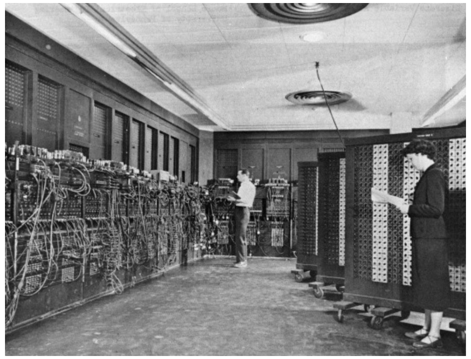
# Number Systems in Computers

- **Binary ( 0 and 1)**
  - Example: computers we are using today
- **Ternary ( -1, 0 , +1)**
  - Example:

Source: https://en.wikipedia.org/wiki/Setun

Setun ternary computer designed by Nikolay Brusentsov in the Soviet Union (1958 Moscow State University)

- **Decimal**
  - Example:



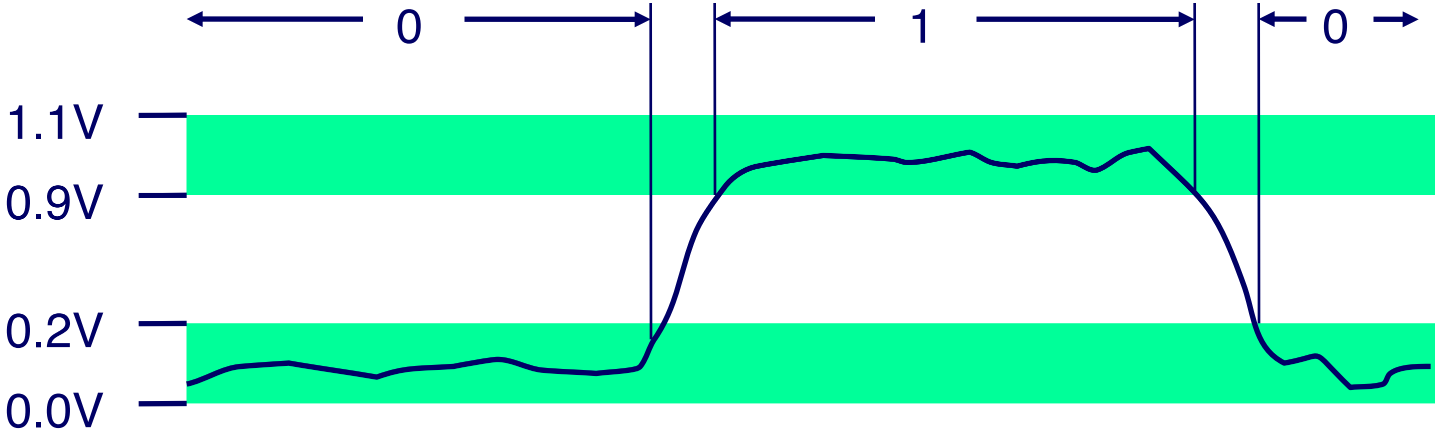
Source: https://en.wikipedia.org/wiki/ENIAC

ENIAC - Designed by John Mauchly and J. Presper Eckert at the University of Pennsylvania, U.S in ~1943.

# Everything is bits

- **Each bit is 0 or 1**
- **By encoding/interpreting sets of bits in various ways**
  - Computers determine what to do (instructions)
  - … and represent and manipulate numbers, sets, strings, etc…
- ~~**Why bits? Electronic Implementation**~~
  - Easy to store with bistable elements
  - Reliably transmitted on noisy and inaccurate wires (are trits not reliable?)

0 1 1 1 0 1 1 0 1 … 1



# For example, can count in binary

- **Base 2 Number Representation**
  - Represent $15213_{10}$ as $11101101101101_2$
  - Represent $1.20_{10}$ as $1.0011001100110011[0011]\ldots_2$
  - Represent $1.5213 \times 10^4$ as $1.1101101101101_2 \times 2^{13}$

# Encoding Byte Values

- **Byte = 8 bits**
  - Binary $00000000_2$ to $11111111_2$
  - Decimal: $0_{10}$ to $255_{10}$
  - Hexadecimal $00_{16}$ to $FF_{16}$
    - Base 16 number representation
    - Use characters ‘0’ to ‘9’ and ‘A’ to ‘F’
    - Write $FA1D37B_{16}$ in C as
      - 0xFA1D37B
      - 0xfa1d37b

**Q: Why a byte is 8-bits ?**

**A: Due to IBM 360 (~1964)**

| Hex | Decimal | Binary |
|---|---|---|
| 0 | 0 | 0000 |
| 1 | 1 | 0001 |
| 2 | 2 | 0010 |
| 3 | 3 | 0011 |
| 4 | 4 | 0100 |
| 5 | 5 | 0101 |
| 6 | 6 | 0110 |
| 7 | 7 | 0111 |
| 8 | 8 | 1000 |
| 9 | 9 | 1001 |
| A | 10 | 1010 |
| B | 11 | 1011 |
| C | 12 | 1100 |
| D | 13 | 1101 |
| E | 14 | 1110 |
| F | 15 | 1111 |

**John von Neumann: “Young man, in mathematics you don't understand things. You just get used to them.”**

Reply, according to Dr. Felix T. Smith of Stanford Research Institute, to a physicist friend who had said "I'm afraid I don't understand the method of characteristics," as quoted in The Dancing Wu Li Masters: An Overview of the New Physics (1979) by Gary Zukav, Bantam Books, p. 208, footnote.

# Example Data Representations (in bytes)

| C Data Type | Typical 32-bit | Typical 64-bit | x86-64 |
|---|---|---|---|
| **`char`** | 1 | 1 | 1 |
| **`short`** | 2 | 2 | 2 |
| **`int`** | 4 | 4 | 4 |
| **`long`** | 4 | 8 | 8 |
| **`float`** | 4 | 4 | 4 |
| **`double`** | 8 | 8 | 8 |
| **`long double`** | – | – | 10/16 |
| pointer | 4 | 8 | 8 |

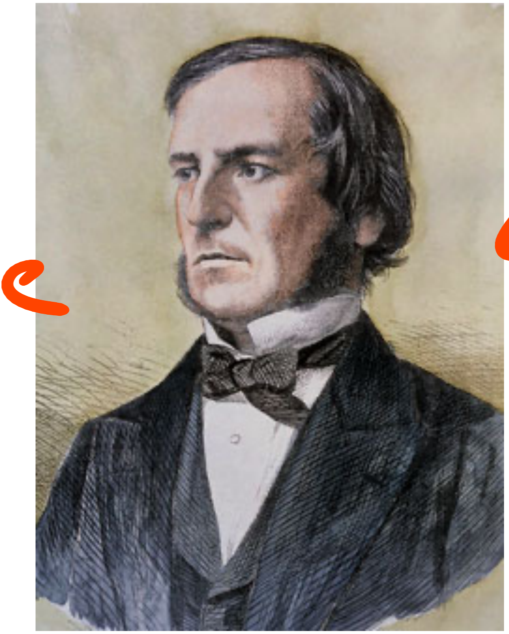
# Today: Bits and Bytes

- Compiling, linking and executing:  Hello  World
- Representing information as bits
- Bit-level manipulations

# Boolean Algebra

-1 , 0 , +1

false maybe true



- **Developed by George Boole in 19th Century**
  - Algebraic representation of logic
    - Encode "True" as 1 and "False" as 0

And

- A&B = 1 when both A=1 and B=1

| & | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Or

- A|B = 1 when either A=1 or B=1

| \| | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

Not

- ~A = 1 when A=0

| ~ | |
|---|---|
| 0 | 1 |
| 1 | 0 |

Exclusive-Or (Xor)

- A^B = 1 when either A=1 or B=1, but not both

| ^ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

# General Boolean Algebras

- **Operate on Bit Vectors**
  - Operations applied bitwise

```
  01101001     01101001     01101001
& 01010101   | 01010101   ^ 01010101   ~ 01010101
  --------     --------     --------     --------
  01000001     01111101     00111100     10101010
```

- **All of the Properties of Boolean Algebra Apply**

# Example: Representing & Manipulating Sets

- **Representation**
  - Width w bit vector represents subsets of {0, …, w–1}
  - $a_j = 1$ if $j \in A$
    - 01101001 { 0, 3, 5, 6 }
    - *76543210*
    - 01010101 { 0, 2, 4, 6 }
    - *76543210*
- **Operations**

| | | | |
|---|---|---|---|
| & | Intersection | 01000001 | { 0, 6 } |
| \| | Union | 01111101 | { 0, 2, 3, 4, 5, 6 } |
| ^ | Symmetric difference | 00111100 | { 2, 3, 4, 5 } |
| ~ | Complement | 10101010 | { 1, 3, 5, 7 } |

# Bit-Level Operations in C

- **Operations &, |, ~, ^ Available in C**
  - Apply to any "integral" data type
    - `long, int, short, char, unsigned`
  - View arguments as bit vectors
  - Arguments applied bit-wise
- **Examples (Char data type)**
  - `~0x41 → 0xBE`
    - `~01000001₂ → 10111110₂`
  - `~0x00 → 0xFF`
    - `~00000000₂ → 11111111₂`
  - `0x69 & 0x55 → 0x41`
    - `01101001₂ & 01010101₂ → 01000001₂`
  - `0x69 | 0x55 → 0x7D`
    - `01101001₂ | 01010101₂ → 01111101₂`

# Contrast: Logic Operations in C

- **Contrast to Logical Operators**
  - `&&`, `||`, `!`
    - View 0 as “False”
    - Anything nonzero as “True”
    - Always return 0 or 1
    - Early termination

Watch out for && vs. & (and || vs. |)… one of the more common oopsies in C programming

- **Examples (char data type)**
  - `!0x41` → `0x00`
  - `!0x00` → `0x01`
  - `!!0x41` → `0x01`

  - `0x69 && 0x55` → `0x01`
  - `0x69 || 0x55` → `0x01`
  - `p && *p` (avoids null pointer access)

P=NuLL

# Shift Operations

- **Left Shift: `x << y`**
  - Shift bit-vector **`x`** left **`y`** positions
    - Throw away extra bits on left
    - Fill with 0's on right
- **Right Shift: `x >> y`**
  - Shift bit-vector **`x`** right **`y`** positions
    - Throw away extra bits on right
  - Logical shift
    - Fill with 0's on left
  - Arithmetic shift
    - Replicate most significant bit on left
- **Undefined Behavior**
  - Shift amount < 0 or ≥ word size

| **Argument x** | ~~011~~00010 |
|---|---|
| << 3 | 00010000 |
| **Log. >>** ~~2~~ 3 | 00001100 |
| **Arith. >>** ~~2~~ 3 | 00001100 |

| **Argument x** | ~~101~~00010 |
|---|---|
| << 3 | 00010000 |
| **Log. >>** ~~2~~ 3 | 00010100 |
| **Arith. >>** ~~2~~ 3 | 11110100 |

*Thank you !*