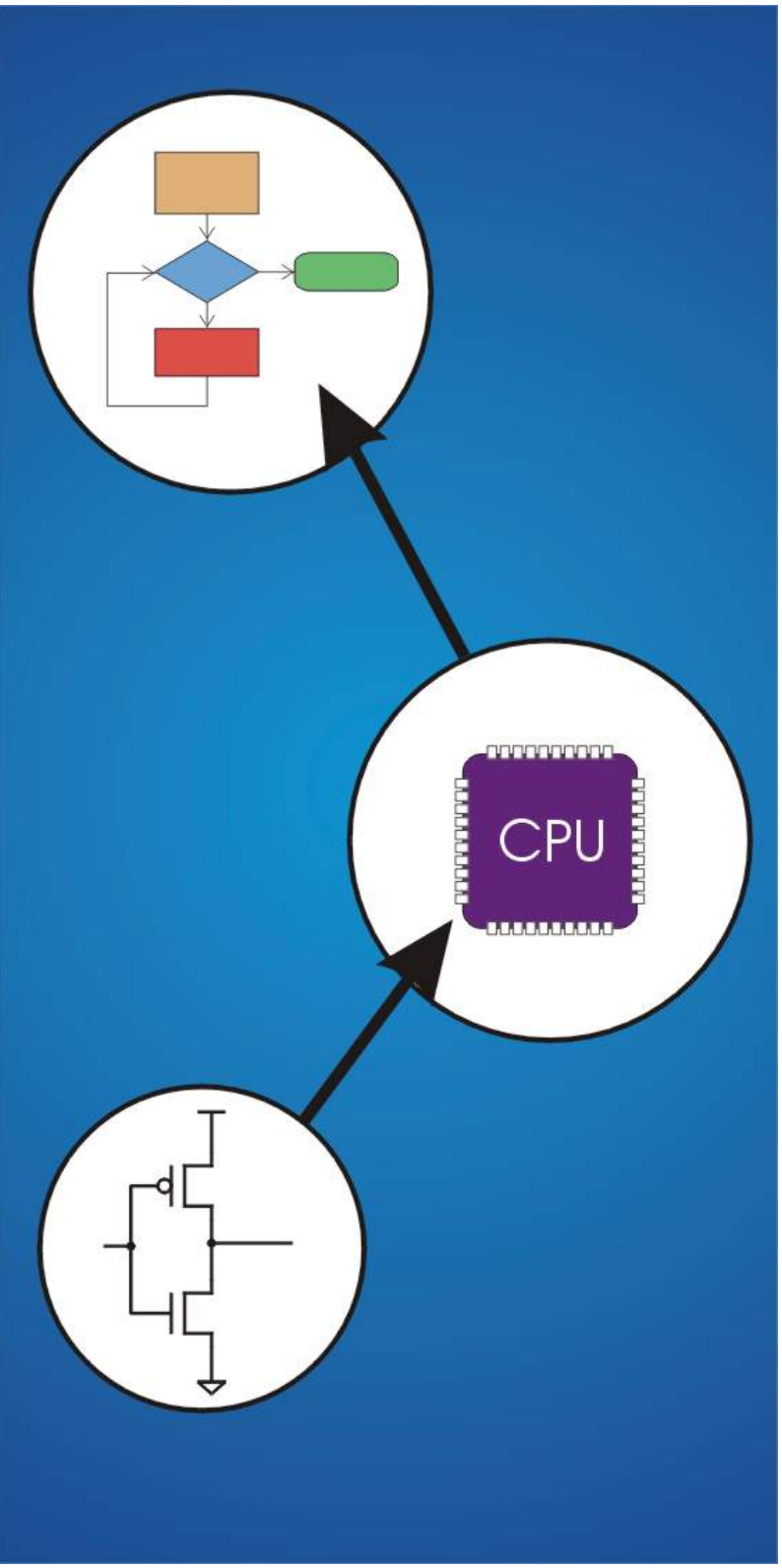


Wrap-up Questions

Course Outline

Bits and Bytes

- How do we represent information using binary representation?

Processor and Instruction Set

- What are then components of a processor?
- What operations (instructions) will we implement?

Assembly Language Programming

- How do we use processor instructions to implement algorithms?
- How do we write modular, reusable code? (subroutines)

I/O, Traps, and Interrupts

- How does processor communicate with outside world?

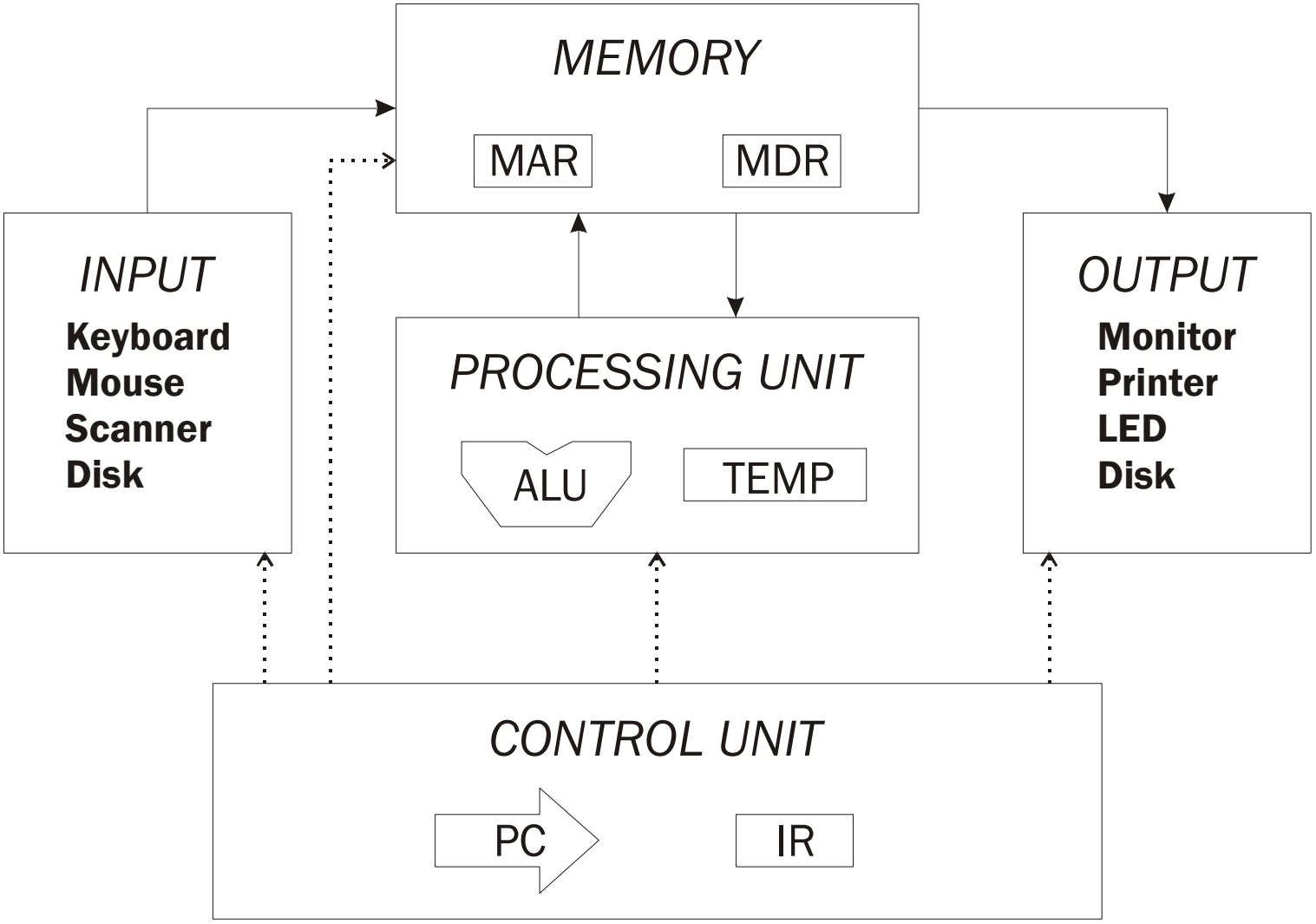
C Programming

- How do we implement high-level programming constructs at the machine level?

Chapter 2: binary representation & data types

- **Why is 2's complement representation used to represent signed integers?**
- **What does this represent: 00100011**
- **Show two ways (in C language) to determine whether a number is even or odd**

Von Neumann Model



Memory and Instructions

- **What is the address space (number of locations) and the addressability (# of bits stored at each location) of the LC-3?**
- **What are the 6 phases of the instruction processing cycle?**

LC-3

- What are the addressing modes of the LC-3?
- LC-3 instructions fall into three classes. What are they?
- What does the first line of an LC-3 machine language or assembly language program contain? Why is it needed?

Tricky assembly language question

What is the value of R1 when the HALT instruction is reached?

```
   .ORIG x3000
   LD R0,A
   AND R1,R1,#0
   STR R1,R0,#3
   ADD R1,R1,#5
   TRAP x25
A .fill X3000
   .END
```

Reverse Assembly

Fill in the blanks:

```
        .ORIG x3000          0011000000000000
        AND R0, R0, x0       0101000000100000
        AND R1, R1, x0       0101001001100000
        ADD R1, R1, x9       0001001001101001
                             0000100000000100
        LD  R2, FF           0010010000001000
        LEA R3, FF            1110011000000111
                             0111001011000010
        LEA R7, DD           1110111000000011
EE      NOT R5, R5           1001101101111111
        BRnz DD              0000110000000001
        NOT R4, R3           1001100011111111
                             0110110010000001
        TRAP x25             1111000000100101
                             1101000000000000
        .FILL xFF00          1111111100000000
        .FILL xFAFA          1111101011111010
        .END
```

I/O

- What does it mean to say that I/O is memory-mapped?
- What is the danger of not testing the DSR before writing data to the screen?
- What determines whether an interrupt signal is sent to the control unit?
- When does the control unit test for an interrupt signal?

TRAP routines and subroutines

- What are TRAP routines and how do they differ from subroutines?
- How do they receive and send information to the caller?
- What needs to be done before one subroutine calls another?

Stacks

- Draw a picture of a software stack as implemented on the LC-3
  - How does one keep track of the top of the stack?
- How is a stack used to service interrupts?

C & LC-3 Assembly Language

- **Compile the following into LC-3 assembly language**

```
int i = 0, j = 2;
i = j + 10;
if(i > 10)
        i = i - 10;
putchar(i);
```

Runtime stack and heap

For each item highlighted in the code below, indicate where that item is allocated in memory. Is it on the run-time stack or is it on the heap?

```
int main(){
  int x;
  int *y;

  x = 3;

  y = (int *) malloc(sizeof(int));

  *y = 4;

  return 0;
}
```

Many More Low Level Programming Opportunities!

- **CS 3413 Operating Systems I**
- **CS 3853 Computer Architecture and Organization**
- **CS 3873 Net-centric Computing**
- **CS 4405 Operating Systems II**
- **CS 4735 Computer Graphics**
- **CS 4745 Introduction to Parallel Processing**
- **Several security courses**

An excellent reference book: *Computer Systems, a Programmer's Perspective*, by Bryant and O'Hallaron , 3rd ed, Prentice Hall