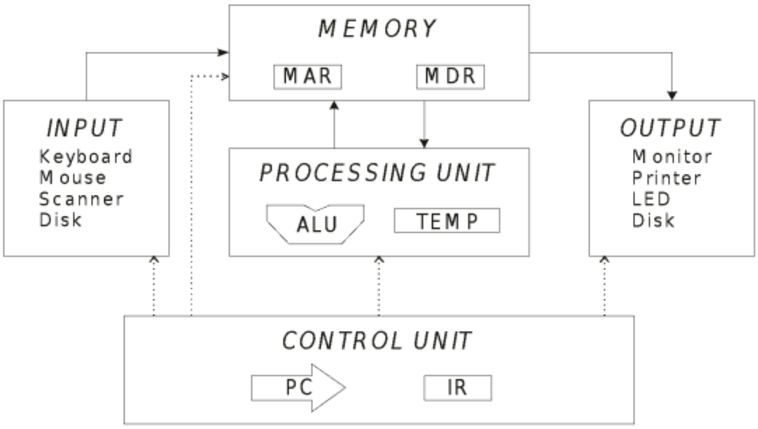
1 Introduction to Computing Systems

- Computers process information
  - take in data/information
  - manipulate or process the data
  - return an output as a result of the processing done
- Elements of information processing
  - Computational ability
  - Memory/data storage
- What programmers do?
  - computer needs a set of instruction (program)
  - programmers write these programs for computers using a language understood by the computer
    - high level language
    - assembly language
    - machine language
- Programming languages
  - High Level
    - closer to human language than machine language
    - needs a compiler or interpreter
    - compiler converts high-level programs into machine code
    - interpreters execute high-level statements/instructions at runtime
    - reflects the elements of computer: computational ability (functions, operations) and memory/data storage (variables, files)
- Programming a computer
  - computers do not ”read between the lines”
  - they need step-by-step instructions
  - higher level language = closer to human language. trade-off: fine control (low level) vs. ease of programming (high level)

2 The Hardware/Software Interface

- Computer. a fast electronic calculating machine that accepts digitized input information, processes it according to a list of internally stored instructions, and produces the resulting output information
- von Neumann Architecture.



- Processor. contains ALU (Arithmetic Logic Unit) and Control Unit/Controller
  - ALU - contain functional units for performing operations on data. has fast memory called registers to contain operands
  - Control unit/controller - the ”conductor” of the processor ”orchestra”
- Memory
  - Stores instructions ...
    - explicit commands that govern transfer of information within and outside the computer
    - specify operations to be performed on (input) data
  - ... and data
    - encoded digital information
    - operands for the intructions
  - Two classes:
    - Primary storage (main memory). fast memory that contains programs or data that are currently executed
    - Secondary storage. cheaper, slower memory that contains programs or data not currently executed or those which are not accessed frequently
    - Trade off. speed (primary) vs size (secondary) vs cost (primary)
- Input/Output. interface to the world (humans and other computers)
- There wasn’t always ”high” and ”low”
  - machine language was used in the past (too tedious)
  - eventually, symbolic notation was developed
  - assembler translates assembly language to machine language
  - assembler also uses necessary addressed
  - assembly is still tedious because programmers have to think at machine level
  - compilers were later developed
- Hardware-software hierarchy
  - Application software
    - perform intended functions or operations of computer
    - usually written in high level language
    - typical programs
  - System software
    - handles basic I/O operations
    - allocates storage and memory
    - provides sharing of computer (and resources) among multiple applications
    - important examples: compiler and operating system

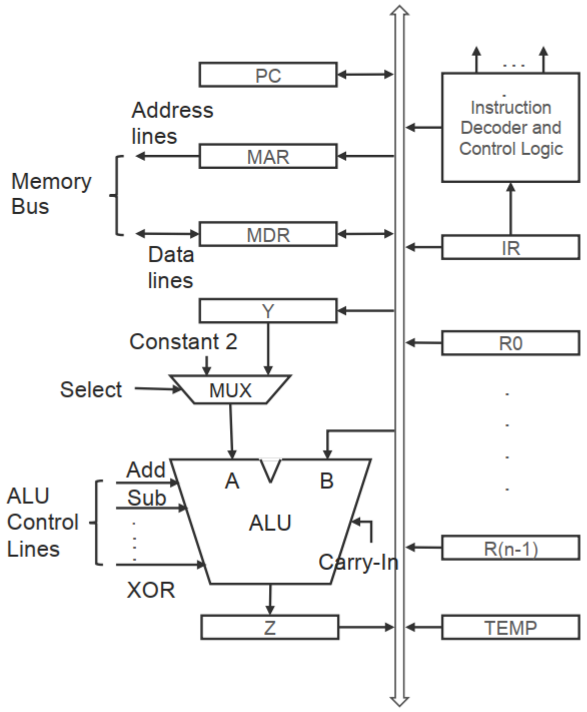
3 Executing Instructions

- How to execute an instruction
  - fetch an instruction from memory
  - fetch the operands
  - execute the instructions
  - store the results
- Memory interaction
  - Control path: instruction fetch and execute cycle; operand fetch; saving results
  - Data path: general purpose registers; ALU; execute instructions

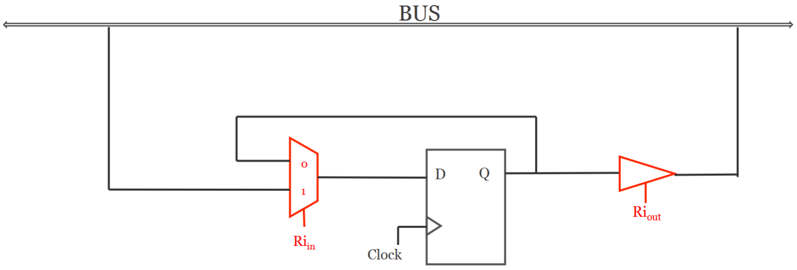
- Fetch - execute cycle
  - Fetching an instruction
    - Fetch instruction from memory location whose address is in the program counter (PC)
    - Place the instruction in the instruction register (IR)
  - Executing an instruction
    - Instruction in the IR is decoded to determine which operation is to be performed
    - Fetch the operands from the memory or registers
    - Execute the operation
    - Store the results in the destination location
- How is memory organized?
  - information is stored as a collection of bits called a **word**
  - a collection of 8 bits is known as **byte**
  - word length can be 16, 32, or 64 bits
  - each byte in memory is assigned an address
    - Little endian - last byte in the word has lowest memory address
    - Big endian - last byte in the word has highest memory address
  - if there are k bits used to hold memory address, there are 2^k addressable space
- Memory operations
  - Memory Read Operation (LOAD)
    - Place the address of the memory location to be read from into **MAR**
    - Issue a **MEM_read** command to the memory
    - Data read from the memory is placed into **MDR** automatically (by control logic)
  - Memory Write Operation (STORE)
    - Place the address of the memory location to be written to into **MAR**
    - Place data to be written into **MDR**
    - Issue a **MEM_write** command to the memory
    - Data in **MDR** is written to the memory automatically (by control logic)
- Types of Instructions/Operations
  - Data transfer between memory and processor registers
  - Arithmetic and logic operations
  - Program sequencing and control flow
  - I/O data transfers
- Operands. operands specify location where data to be used by an instruction is located (memory location or register). may be the source or destination of information
- Addressing modes for operands
  - Register mode − operands in register locations
  - Absolute mode − operands in a memory location
  - Immediate mode − operand is given explicitly in the instruction
  - Indirect mode − the effective address of an operand is contents of a register
  - Index mode − effective address is generated by adding a constant value to the contents of the register
  - Autoincrement − the effective address of the operands is the content of the register. afterwards, the content of the register is incremented
  - Autodecrement − the effective address of the operands is the content of the register. afterwards, the content of the register is decremented
  - Note for autoincrement/autodecrement: $+(\mathrm{R})$ or $-(\mathrm{R})$ increase/decrease before use; $(\mathrm{R})+$ or $(\mathrm{R})-$ increase/decrease after use

4 From Instructions to Control Signals

- Control unit. it is the physical entity in a processor that:
  - Fetches instructions
  - Fetches operands
  - Decodes instructions
  - Schedules events in the data path for the instruction to be executed
  - This is repeated until all instructions are executed
- Fetch / Execute cycle
  - Step 1
    - Fetch the contents of the memory location pointed to by the **PC**
    - PC points to the memory instruction which has the instruction to be executed
    - Load the contents of the memory location into **IR**
  - Step 2
    - Increments the content of the PC by word length
  - Step 3
    - Carry out instruction specified by the instruction in the IR
  - Steps 1 and 2 are Fetch; Step 3 is Execute
  - how are processor units organized and how they communicate? BUS
- Single bus organization



- Register and the BUS
  - bus can be viewed as a collection of parallel wires
  - buses have no memory
  - when a data is on bus, all registers can ”see” that data at their inputs
  - a register may place its contents onto the bus
  - only one register can place its output on the bus at any one time
  - which registers place data or load data from the bus is determined by the control signal issued by the control logic



  - registers are clocked (sequential) entities

- Registers
  - each bit in register may be implemented by an edge triggered D flip flop
  - two input multiplexer **MUX** is used to select the data applied to the input of an edge triggered flip-flop
  - Q output of the flip-flop is connected to the bus via a tri-state gate
  - $R_{\text{i, in}} = 1$: multiplexer selects data on bus; data is loaded into the flip-flop on the rising edge of clock
  - $R_{\text{i, in}} = 0$: MUX feeds back the value currently stored in the flip-flop; Q output represents the value currently stored
  - Tri-state gate. can have the following input states
    - logic low (L)
    - logic high (H)
    - open-circuit (high impedance, Z)

- Clock signal
  - data is loaded at the L−H transition of the clock
  - data transfer and operations take place within time periods called **clock cycle**
  - control signals that govern a transfer are asserted at beginning of clock cycles
  - edge-triggered flip-flop uses only the rising edge
  - when edge-triggered flip-flops are not used, two or more clocks may be used to guarantee proper data transfer (called **multiphase clocking**)

5 MARIE: a Simple Processor

- MARIE: **M**achine **A**rchitecture that is **R**eally **I**ntuitive and **E**asy