Introduction

In a Computer…

- Memory is separate from the CPU
- All data is in binary (not decimal)

Components of a CPU

- **ALU (arithmetic and logic unit)**
  - Perform arithmetic and logic operations
  - Arithmetic: add, subtract, multiply, divide, etc.
  - Logic: AND, OR, NOT, Shift, etc.
- **Control unit**
  - Interprets instructions
  - Controls the flow of information within the CPU
  - Works with a “program counter” (address of next instruction)
- **Input/output interface**
  - Provides mechanism for input and output of data
  - Many variations possible
Program Counter

- Sometimes called instruction pointer.
- Contains address of current or next instruction.
- Usually increased automatically at end of instruction cycle, but may be changed deliberately by program.
- Changes in PC create branches in program.

(See also: Slide # 22)

Registers

- A register is a storage location within the CPU
- Unlike memory, which is “outside” the CPU
- Examples of registers:
  - Accumulator (ACC)
  - Program counter (PC)
  - Instruction register (IR)
  - Memory address register (MAR)
  - Memory data register (MDR)
  - Status register
- General purpose registers (R0, R1, …)
  - Included on some CPUs
  - Used for high-speed temporary storage

Memory Unit

- 2^n x m
- n address bits = 2^n addresses
- m data bits
- m is the “width” of the data path
- Typical values:
  - n: 16, 17, 18, 19, 20, 21, 22, etc.
  - m: 8, 16, 32, 64
Question

• Q: How many bits of memory are contained in a memory unit with 512KB of memory?

A: \(512 = 2^9, K = 2^{10}, B = \text{byte} = 8 = 2^3\)

\[2^9 \times 2^{10} \times 2^3 = 2^{22} = 4,194,304\]

Exercise – Memory Capacity

• Q: How many bits of memory are contained in a memory unit with 2MB of memory?

A:

Answer

Exercise – Memory Capacity

• Q: How many bits of memory are contained in a memory unit with 2MB of memory?

A: \(2 = 2^1, M = 2^{20}, B = \text{byte} = 8 = 2^3\)

\[2^1 \times 2^{20} \times 2^3 = 2^{24} = 16,777,216\]

Memory Implementations

• RAM – random access memory
  • Static RAM
  • Dynamic RAM
• ROM – read-only memory
  • EPROM – erasable programmable ROM
  • Flash ROM
A dedicated register in the CPU. 
Contains the address in memory of the current instruction being executed. 
Incremented automatically after each instruction. 
May be forced to change: eg “jump” instruction. 
Usually initialize to zero when machine starts, or is reset.

**Program Counter (PC)**

A dedicated register in the CPU which contains the actual current instruction.

**Instruction Register (IR)**

Contains Address in memory to find or place data.

**Memory Address Register (MAR)**

Contains Address in memory to find or place data.

**Memory Data Register (MDR)**

Contains Actual Data to be placed in location given in MAR, or which has been retrieved from location given in MAR.

**Instruction Register (IR)**

A dedicated register (set of registers) in the CPU used for the actual manipulation of data.

Default source (or destination) register.

Usually contains results of arithmetic or logical operations.
Generic CPU With Registers

- Program Counter (PC)
- Instruction Register (IR)
- Memory Address Register (MAR)
- Memory Data Register (MDR)
- Accumulator (A or Acc)

Fetch-Execute Cycle

- Two steps, or cycles, in the execution of every instruction
  - Fetch – fetch the code for the instruction from memory and place it in the IR (instruction register)
  - Execute – execute the instruction
- A pretty picture sometimes helps…

Fetch: PC → MAR
MDR → IR
IR[address] → MAR

Execute: A → MDR
PC + 1 → PC

The Store Instruction

Fetching: PC → MAR
MDR → IR
IR[address] → MAR

Executing: A → MDR
A + MDR → A
PC + 1 → PC

The Add Instruction

Fetching: PC → MAR
MDR → IR
IR[address] → MAR

Executing: A + MDR → A
PC + 1 → PC
Fetch-Execute Example: Load Accumulator

Assume: Simple Eight bit system.
Thirty-two memory locations (0 to 31).
“Load” instruction is 010.
Value in location 15 is ten (ie: binary 00001010)
PC is at 5, about to increment.
The instruction, 01001111, is in location 6.

Then ...

CPU Before PC increments

Location 31
15: 00001010
06: 01001111
Location 0

Increment PC: PC = PC + 1

Location 31
15: 00001010
06: 01001111
Location 0

MAR loaded with PC: PC -> MAR

Location 31
15: 00001010
06: 01001111
Location 0
Memory Location 00110 Accessed and Contents Placed in MDR:

PC: 00110
IR: (previous)
MAR: 00110
MDR: (previous)
A: (previous)

Location 0

MDR copied to IR: MDR -> IR

Location 31

MDR copied to IR: MDR -> IR

Location 31

IR [address part] -> MAR

Location 31

IR [address part] -> MAR

Location 31
Location in MAR (01111) Accessed

Location 31

PC: 00110
IR: 01001111
MAR: 01111
MDR: 01001111
A: (previous)

15: 00001010

06: 01001111
Location 0

Contents of 01111 loaded into MDR

Location 31

PC: 00110
IR: 01001111
MAR: 01111
MDR: 00001010
A: (previous)

15: 00001010

06: 01001111
Location 0

IR [op code] executed: MDR -> A

Location 31

PC: 00110
IR: 01001111
MAR: 01111
MDR: 00001010
A: 00001010

15: 00001010

06: 01001111
Location 0

Finished!
Now:
Assume: Value in location 7 is 10110010.
“Add” instruction is 101.
Value in location 18 is seventy-one
(ie: binary 01000111)
Everything else is as we left it!

Then ...

PC = PC + 1

Location 31
18: 01000111
15: 00001010
07: 10110010
06: 01001111
Location 0

PC -> MAR

Location 31
18: 01000111
15: 00001010
07: 10110010
06: 01001111
Location 0

MAR Accesses Location 00111

Location 31
18: 01000111
15: 00001010
07: 10110010
06: 01001111
Location 0
Contents of [10010] -> MDR

Location 31

PC: 00111
IR: 10110010
MAR: 10010
MDR: 01000111
A: 00001010

To Continue:

If the next instruction were to load the Accumulator contents into an area of memory reserved for screen output (for example), then the number “81” should appear on the screen.

The process continues in the same fashion, more or less, until a stop or halt instruction is encountered.

IR [opcode] executed: A = A + MDR

Location 31

PC: 00111
IR: 10110010
MAR: 10010
MDR: 01000111
A: 01010001

Buses

• Definition: a collection of electrical conductors (eg: wires, traces) with a common purpose
• Each wire or trace is called a line
• Typically, buses carry information from one place to another

From our first lecture…
Types of Buses (1 of 3)

- Point-to-point

Types of Buses (2 of 3)

- Multipoint

Buses Inside a Computer
Data Bus

- Carries data between the CPU and memory or I/O devices
- Bi-directional
  - Data transferred “out of” the CPU for write operations
  - Data transferred “into” the CPU for read operations
- Typical sizes: 8, 16, 32, 64 lines
- Signal names:
  - D0, D1, D2, D3, etc.

Address Bus

- Carries an address from the CPU to Memory or I/O devices
- Unidirectional
  - The address is always supplied by the CPU
  (There is one exception to this, which we’ll discuss later.)
- Typical sizes: 16, 20, 24 lines
- Signal names:
  - A0, A1, A2, A3, etc.

Control Bus

- Collection of signals for coordinating CPU activities
- Each signal has a unique purpose
- Typical sizes: 10-20 lines
- Signals are output, input, or bi-directional
- Typical signals
  - /RD (read)
  - /WR (write)
  - CLK (clock)
  - /IRQ (interrupt request)
  - etc.