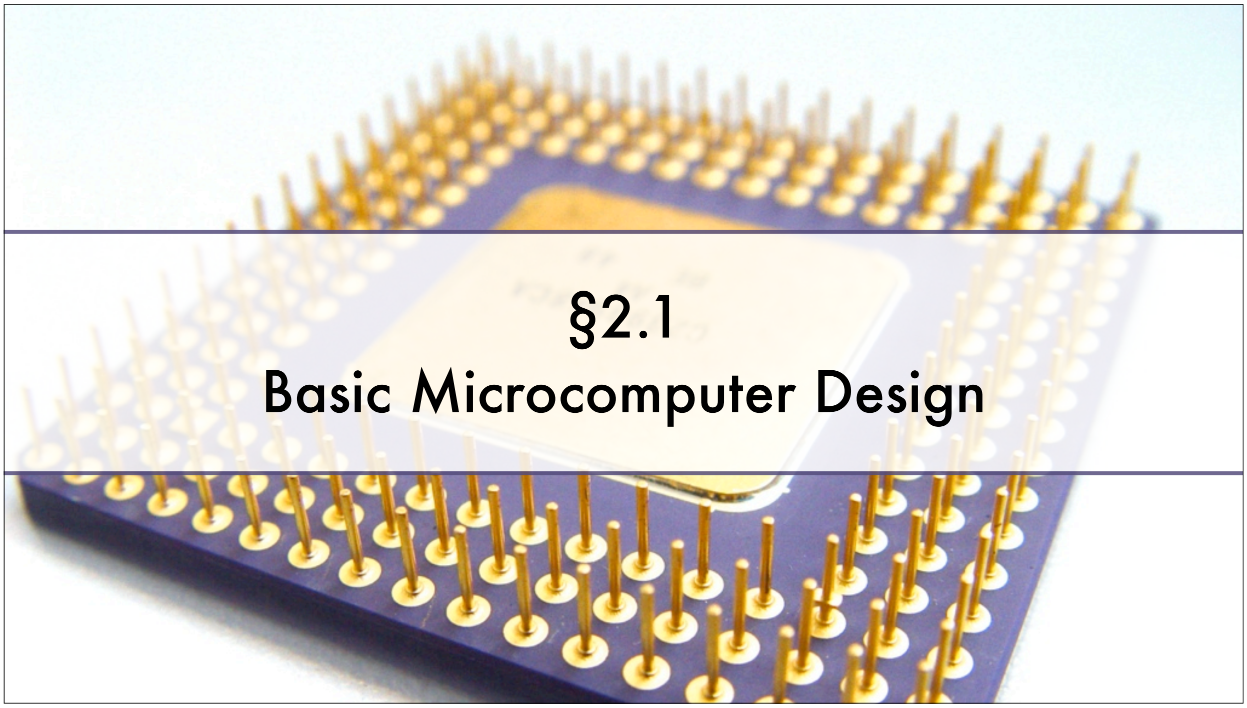
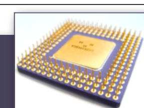


§2.1
Basic Microcomputer Design

Activity 3

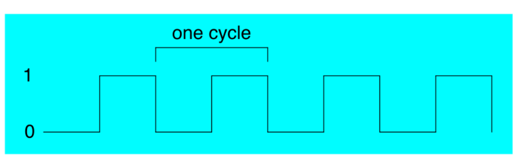


- Complete **Activity 3** now. Fill in the blanks using words from the word banks provided.

Activity 3, Question 1

Clock

- The *clock* pulses at a constant rate and is used to synchronize the internal operations of the CPU with other system components



  - *Clock cycle:* time between consecutive falling edges
  - 1 GHz = 1,000,000,000 cycles per second, so
    each cycle is $^{1}/_{1,000,000,000}$ s = $^{1}/_{10^9}$ s = 10^{-9} s = 1 ns (one nanosecond)

Clock

- **Question:**
  If a clock oscillates at 2.2 GHz, what is the duration of one clock cycle in nanoseconds?
- **Answer:**

2.2 GHz $= 2.2 \cdot 10^9$ Hz, so one clock cycle is $\frac{1}{2.2 \cdot 10^9}$ seconds. To convert to nanoseconds (recall that 1 ns $= 10^{-9}$ s), solve:

$$\begin{aligned} \frac{1}{2.2 \cdot 10^9} &= x \cdot 10^{-9} \\ 10^9 \cdot \frac{1}{2.2 \cdot 10^9} &= x \cdot 10^{-9} \cdot 10^9 \\ \frac{1}{2.2} &= x = \frac{5}{11} \end{aligned}$$

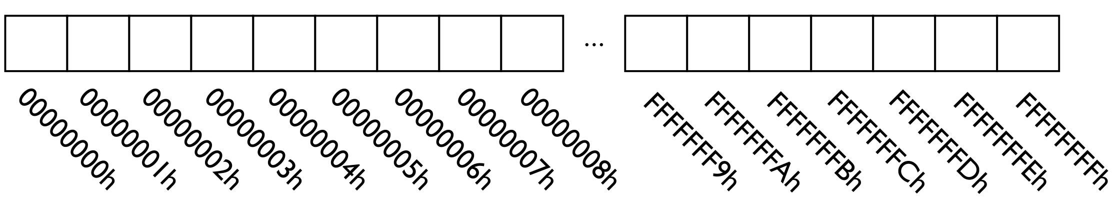
So one clock cycle is $\frac{5}{11} \approx 0.4545$ ns.

Activity 3, Question 2

CPU

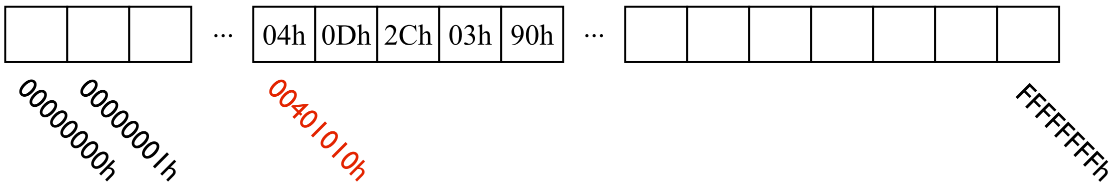
- The *central processing unit (CPU)* contains circuitry to carry out the instructions of a computer program, performing basic arithmetic, logical, and I/O operations. It contains:
  - *ALU* - contains circuitry to perform arithmetic, comparison, and logical operations
  - *Control unit* - contains circuitry to fetch an instruction, decode it, and direct the ALU to carry out the appropriate operation
  - *Registers* - memory locations located inside the CPU that hold intermediate values for computations and are accessed much faster than conventional memory

Memory Storage Unit



- *Memory storage unit* (“memory”) stores both
  - **Instructions** – the machine language code for every all of the programs you have open
  - **Data** – strings, arrays, etc. used by those programs
- You can think of your computer’s memory as a giant array of bytes
  - On a 32-bit processor, the bytes are numbered from 00000000h to FFFFFFFFh
  - This number is the *memory address* of that particular byte
- **Q.** How much memory can be addressed if a processor uses 32-bit memory addresses?

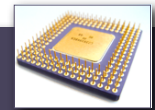
Instruction Pointer



- Remember: Machine language instructions are stored in memory
- The processor has a special register called the *instruction pointer* or *program counter*
  - In x86 processors, this register is named EIP
- This contains the memory address of the next machine language instruction to execute
- So how does the processor actually run a program? Suppose EIP contains 401010h...

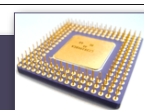
Section 2.1.5, Questions 4–5

FDX Cycle



- A CPU executes a program by repeating the *fetch-decode-execute (FDX)* cycle:
  1. **Fetch**
  2. **Decode**
  3. **Fetch operands** *(memory access instructions only)*
  4. **Execute**
  5. **Store output** *(memory access instructions only)*

FDX Cycle



- A CPU executes a program by repeating the *fetch-decode-execute (FDX)* cycle:
  1. **Fetch** - control unit retrieves the next instruction (i.e., byte(s) of machine code) from memory and increments the *instruction pointer/program counter*
  2. **Decode**
  3. **Fetch operands** *(memory access instructions only)*
  4. **Execute**
  5. **Store output** *(memory access instructions only)*

FDX Cycle

- A CPU executes a program by repeating the *fetch-decode-execute (FDX)* cycle:
  1. **Fetch**
  2. **Decode** - control unit determines what those bytes mean, passes operands to the ALU, sends signals to the ALU indicating what action to take
  3. **Fetch operands** *(memory access instructions only)*
  4. **Execute**
  5. **Store output** *(memory access instructions only)*

FDX Cycle

- A CPU executes a program by repeating the *fetch-decode-execute (FDX)* cycle:
  1. **Fetch**
  2. **Decode**
  3. **Fetch operands** *(memory access instructions only)* - control unit copies operands from memory into internal registers
     - *Internal* registers cannot be accessed by user programs
  4. **Execute**
  5. **Store output** *(memory access instructions only)*

FDX Cycle

- A CPU executes a program by repeating the *fetch-decode-execute (FDX)* cycle:
  1. **Fetch**
  2. **Decode**
  3. **Fetch operands** *(memory access instructions only)*
  4. **Execute** - ALU performs the operation and sets *flags* (in a special *flags* register) indicating whether certain conditions occurred (overflow, carry, etc.)
  5. **Store output** *(memory access instructions only)*

FDX Cycle

- A CPU executes a program by repeating the *fetch-decode-execute (FDX)* cycle:
  1. **Fetch**
  2. **Decode**
  3. **Fetch operands** *(memory access instructions only)*
  4. **Execute**
  5. **Store output** *(memory access instructions only)* - control unit copies result from register to memory

FDX Cycle

- The FDX cycle in pseudocode:

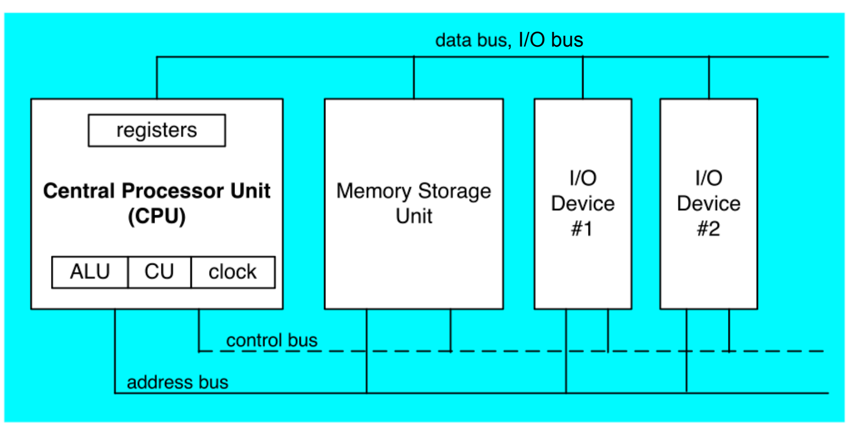
  while (true) {
  Fetch next instruction
  Advance instruction pointer (IP)
  Decode instruction
  If memory operand needed, read value from memory
  Execute instruction
  If output operand is a memory operand, write to memory
  }

- When the CPU is started/reset, the instruction pointer is set to a predefined value
  - E.g., the Intel 8086 processor begins executing instructions at memory address FFFF0h. The machine code at this address is in read-only memory (ROM) and begins the computer's boot sequence.
  - Yes, FFFF0h is a 20-bit memory address, not 32-bit (what?!). Wait for Section 2.3.

Activity 3, Question 3
Section 2.1.5, Question 2

Buses

- A *bus* is a group of parallel wires that transfer data from one part of the computer to another
- The CPU is connected to the rest of the computer system using four buses:
  - Address bus
  - Data bus
  - Control bus
  - I/O bus

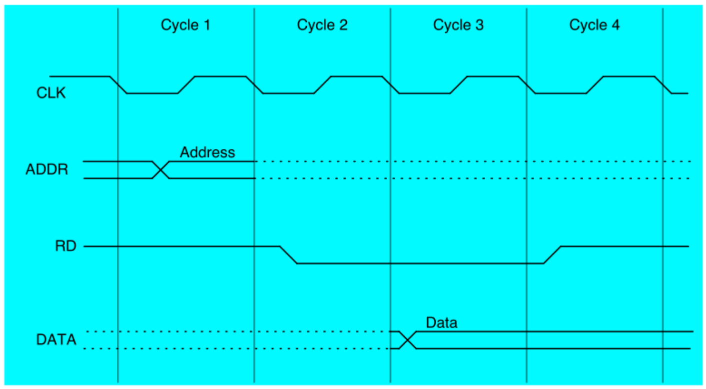


Activity 3, Question 4

Reading from Memory

1. CPU places an address on the address bus
2. CPU's control unit signals the memory storage unit that a read operation should be performed
3. Memory storage unit reads the data from memory and places the data on the data bus
4. CPU's control unit reads the data from the bus and stores it in an internal register in the CPU

Reading from Memory



Cache Memory

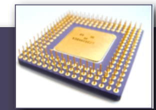
- Accessing conventional memory is *slow,* so...
- When memory is accessed the first time, it is copied into a high-speed *cache memory*
  - Cache memory is faster than conventional memory
  - But it is much smaller: it only contains a copy of a few recently-accessed portions of main memory
- Memory accesses look in the cache first, then access main memory only if necessary
  - *Cache hit:* data found in cache
  - *Cache miss:* data not in cache

Activity 3, Questions 5–6

Operating Systems

- An *operating system (OS)*
  - Manages hardware resources
  - Provides services to application programs, e.g.,
    - Input/output – remember what *device drivers* do? (assigned reading, §1.1)
    - Read/write/delete files
    - Allocate memory
    - Run another program
  - The operating system's *application programming interface (API)* provides functions that you can call from C/C++ or assembly language programs to perform these tasks. Example from the Win32 API: `WriteConsole` (link)
- The part of the OS the user interacts with (e.g., to start programs) is the *shell*

Homework



- For next class (Friday, August 29):
  - **Lab 1** (from Monday) is due <u>on paper</u> at the beginning of class
  - Read **Section 2.2** from the <u>6th edition</u> (pp. 36–42) – PDF in Canvas
  - Be prepared to verbally answer review questions
    1, 2, 3, 6, and 12 from Section 2.2.5