

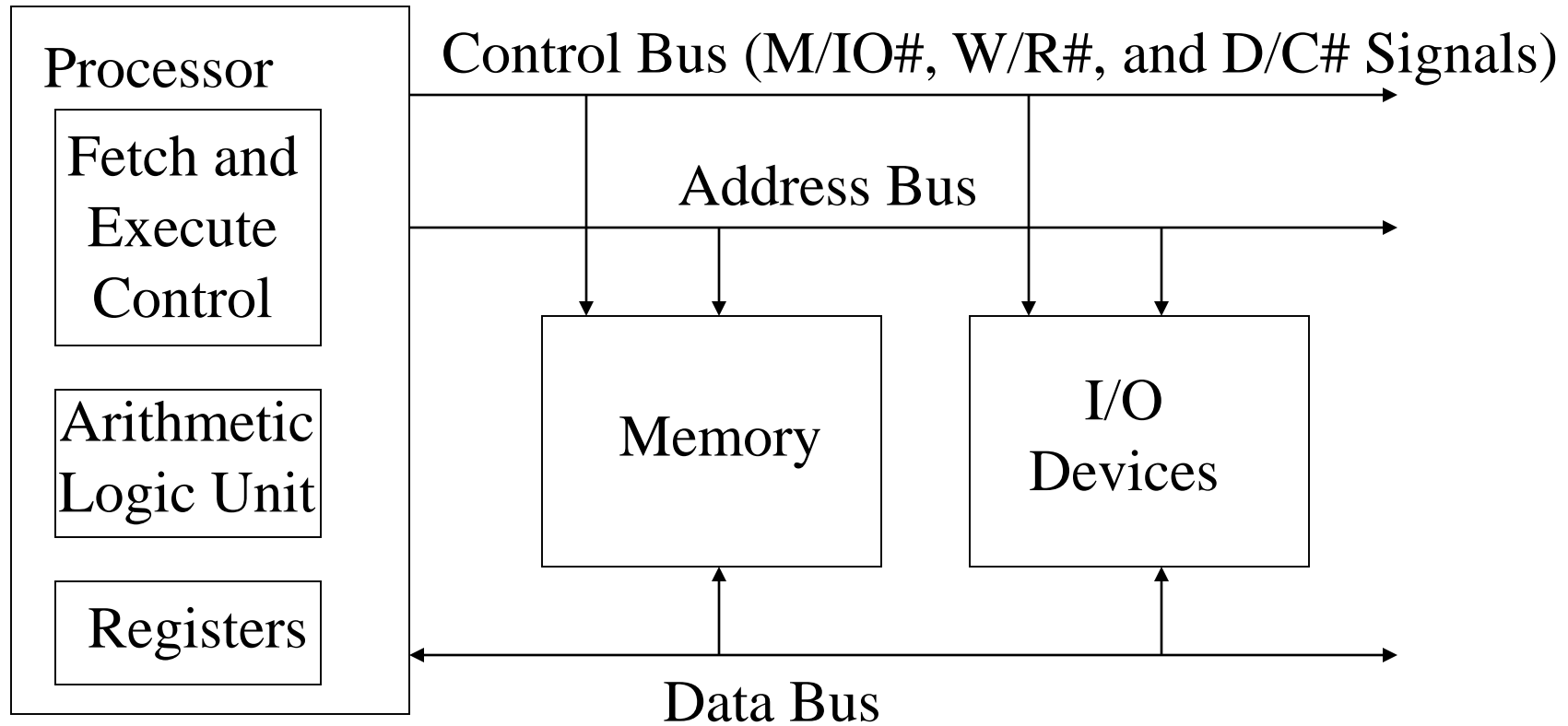
Homework

- Reading
 - Professional Assembly Language, pp 17-32, 34-37
 - Selecting Intel Registers
<http://www.swansontec.com/sregisters.html>
- Continue work on mp1
 - Questions?
- Lab with your assigned section this week

General Computer Architecture

- Processor
 - Controls fetching and execution of instructions
 - Moves data between memory, registers, the arithmetic / logic unit (ALU), and I/O devices
- Memory
 - Stores instructions and/or data
- I/O Devices
 - Bring data into system
 - Send data out from system

General Computer Architecture



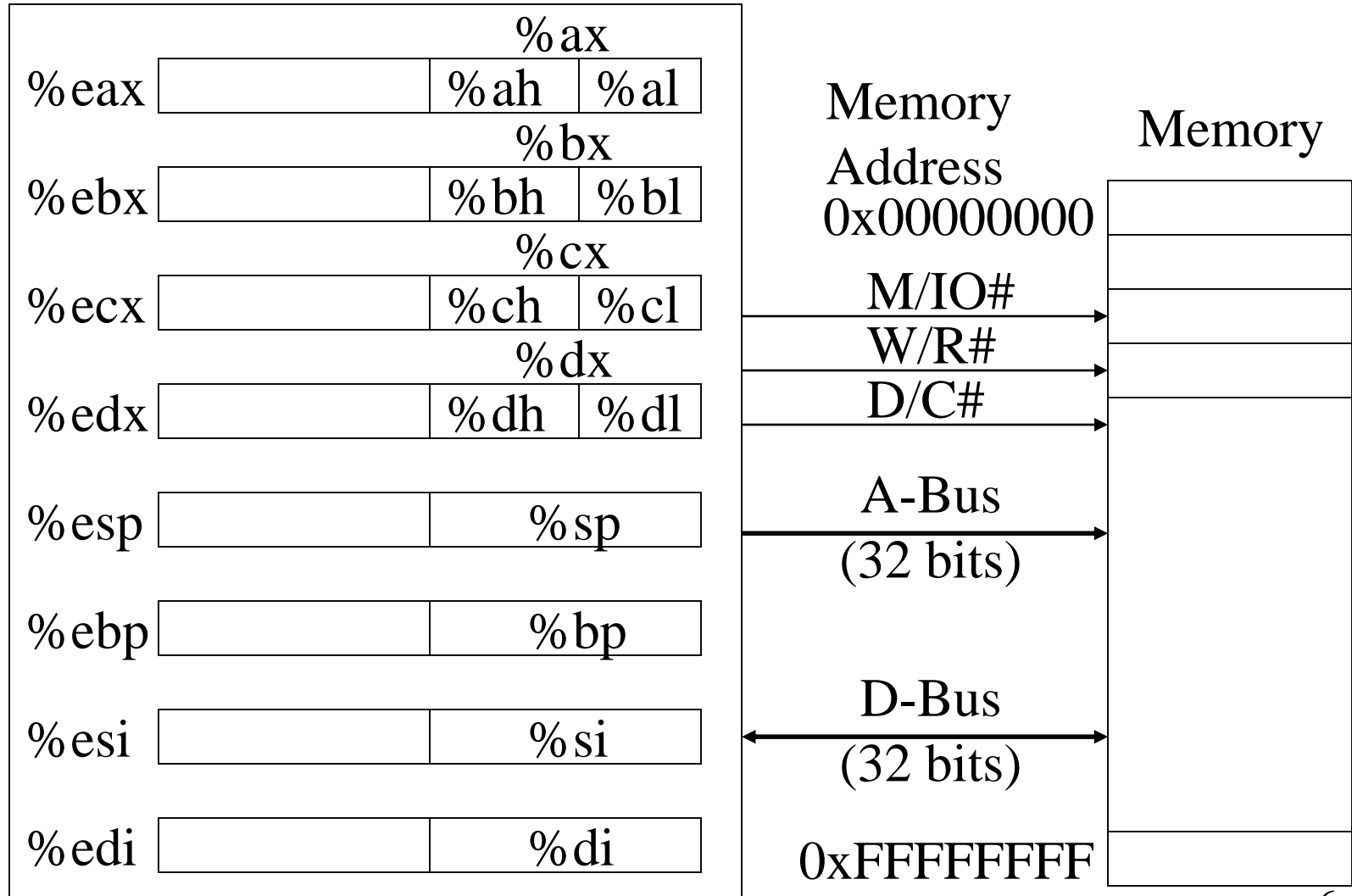
Signals and Busses

- A “signal” is a logical value represented as a voltage on a wire inside the machine
- A signal is binary (two states – on and off)
- There is a specific meaning assigned to each value of a signal, e.g. M/IO#
 - M/IO# = 1 means access memory
 - M/IO# = 0 means access an I/O device
- A “bus” is a group of signals with one purpose

Processor Model

- An assembly language programmer usually thinks of the processor in terms of its:
 - Registers
 - Arithmetic Logic Unit (ALU)
 - Instructions
 - Address and data bus sizes
- I'll be simplifying the textbook material for now
- We'll come back to that material later

80386 Processor Model - Registers



Processor Model - Registers

- Additional status and control registers
 - Instruction Pointer/Extended Instruction Pointer

%eip

| | |
|--|--|
| | |
|--|--|

- Extended Flags Register

%eflags

| | |
|--|--|
| | |
|--|--|

Arithmetic Logic Unit

- Capable of performing arithmetic
 - Addition, Subtraction, Multiplication, Division
- Capable of performing logic operations
 - and, or, exclusive or
- Takes operands from source(s) specified in the instruction
- Delivers results to destination specified in the instruction

Processor Model - Instructions

- Instructions to move a constant into a register
 - `movb $54, %al` Move 54_{10} to al register
 - `movb $0x36, %al` Move 36_{16} to al register
 - `movb $'6', %al` Move digit 6 to al register
- Instructions to move data between registers
 - `movl %ebx, %eax`
 - `movw %bx, %ax`
 - `movb %bh, %ah`
 - `movb %bl, %al`

Processor Model - Instructions

- Instructions to add or subtract a constant to a register

```
addb $10, %bl
```

```
subb $10, %bl
```

- Instructions to add or subtract a register to another register

```
addb %bh, %bl
```

```
subb %bh, %bl
```

Busses

- Address Bus
 - Driven by the processor (Simplified for now)
 - Processor presents address of memory location or I/O device being accessed on the busses
 - Each memory or I/O device determines if the value on the address bus and M/IO# signal selects it or not
 - When selected, memory or I/O device gets data from data bus or puts data on data bus based on W/R#

Busses

- Data Bus
 - Used as a conduit for data between devices
 - Specific operations performed on the data bus are driven by control and address bus signals
 - Can be driven by processor, memory, or an I/O device depending on the type of transfer done
 - Interfaces to data bus require a special kind of “tri-state” logic which we will discuss later

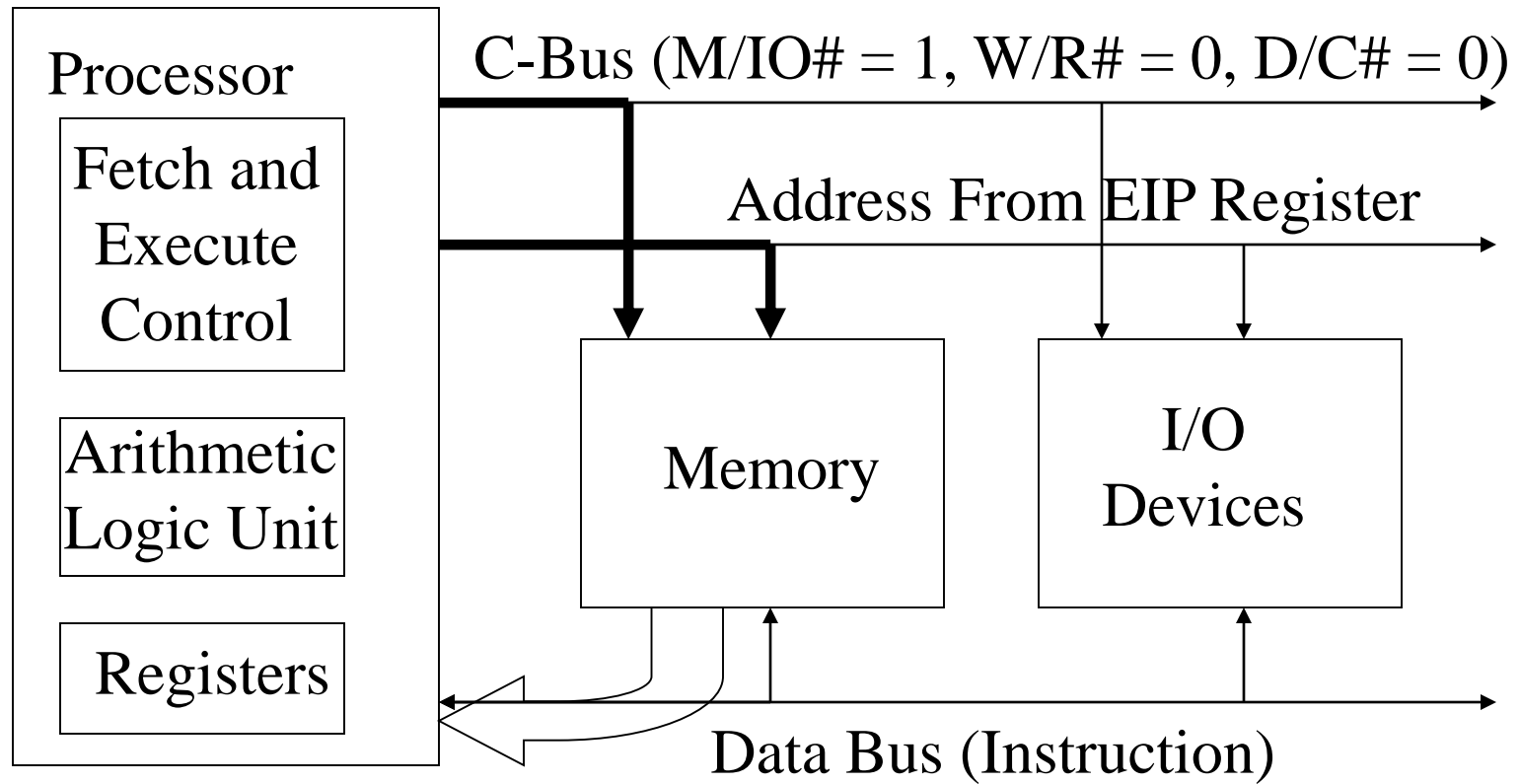
Busses

- Control Bus Signals
 - M/IO# signal selects
 - Memory locations when set (= 1)
 - I/O device registers when reset (= 0)
 - W/R# signal moves data from
 - Processor to memory or I/O device when set (= 1)
 - Memory or I/O device to processor when reset (= 0)
 - D/C# signal indicates
 - Data (instruction execute phase) when set (=1)
 - Control (instruction fetch phase) when reset (=0)

Fetch Cycle

- On each fetch cycle, processor
 - Puts signal $M/IO\# = 1$ on control bus
 - Puts signal $W/R\# = 0$ on control bus
 - Puts signal $D/C\# = 0$ on control bus
 - Puts address of next instruction from the EIP register on address bus signals
 - Reads next instruction on data bus

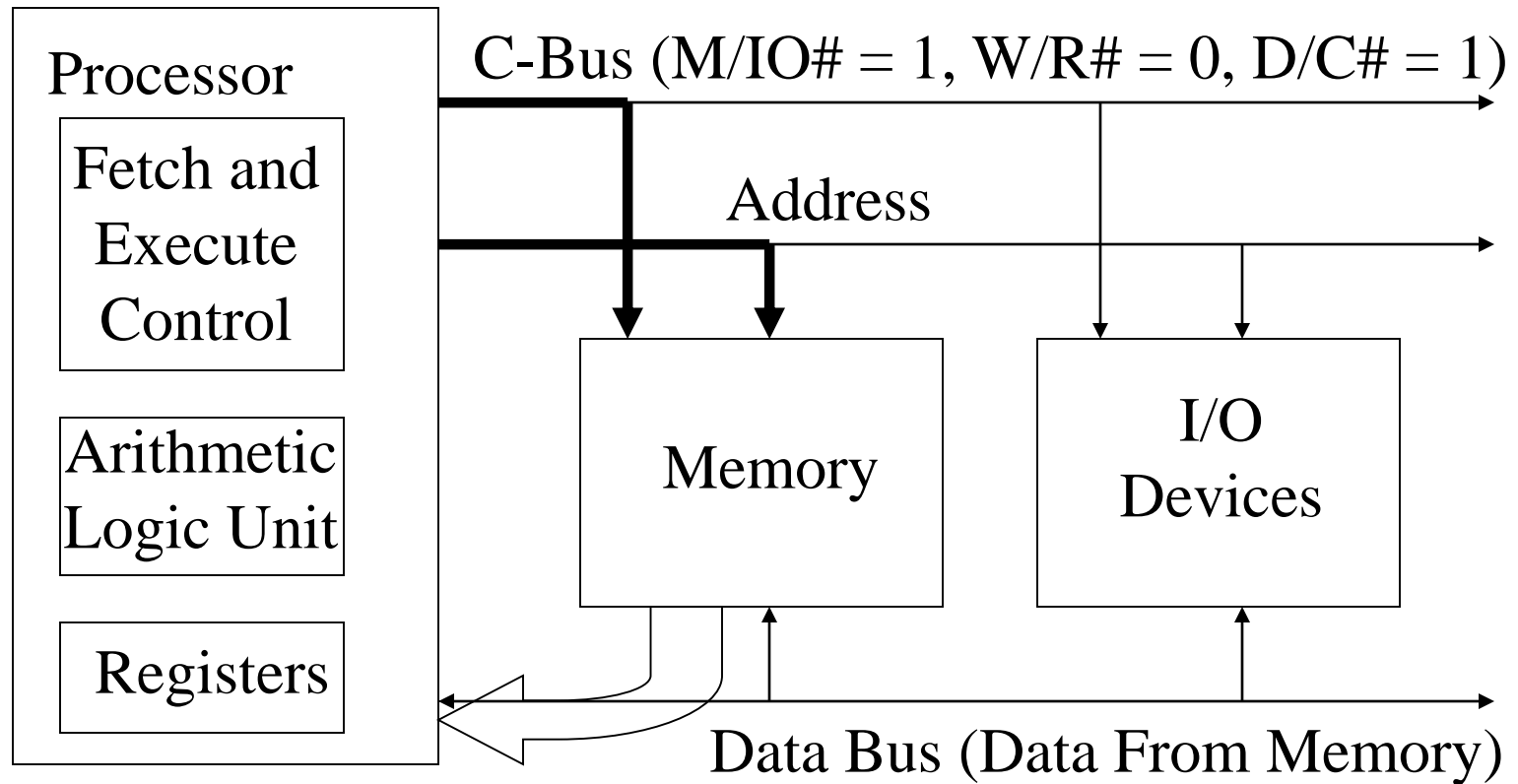
Fetch Cycle



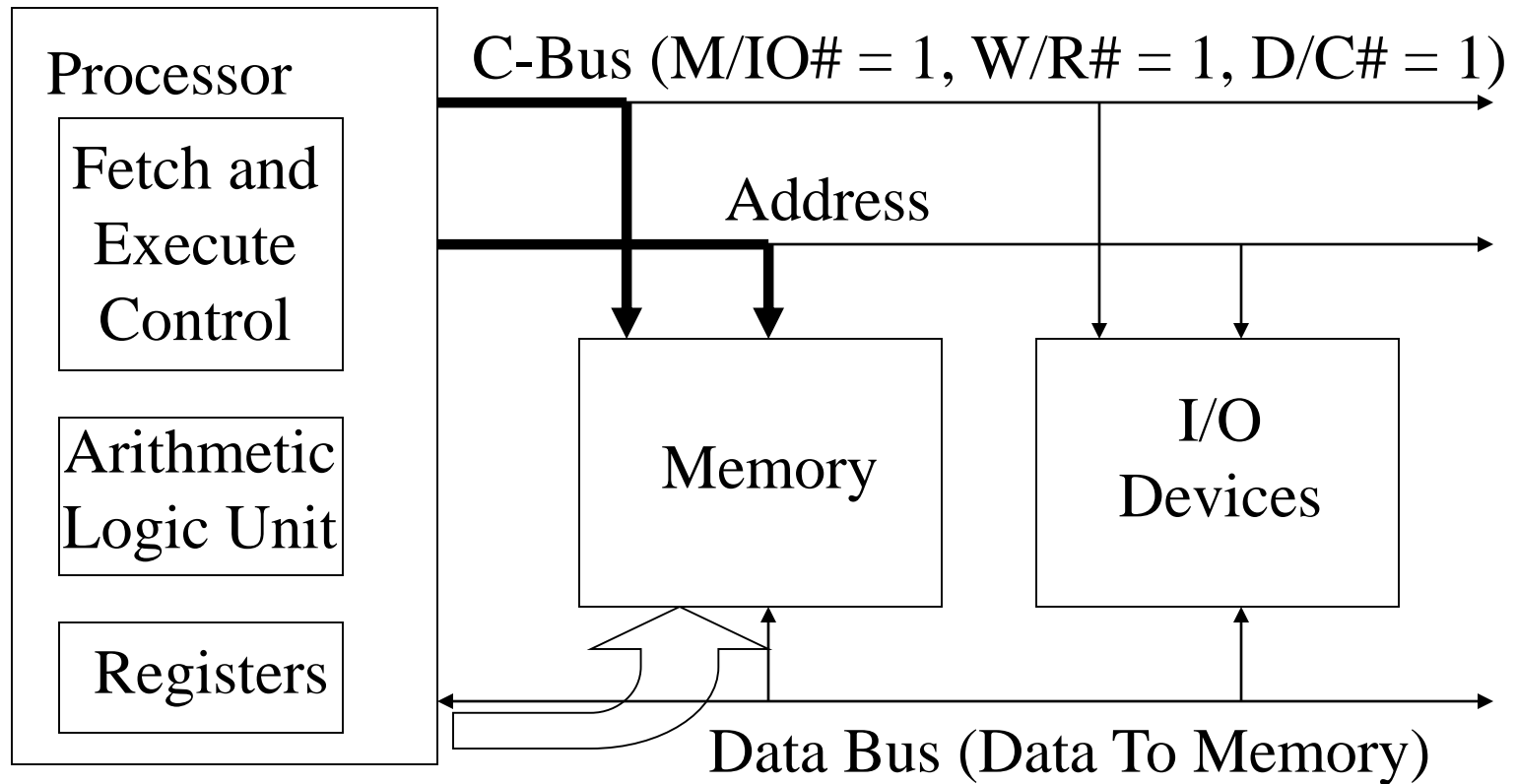
Execute Cycle

- On each execute cycle, processor
 - May or may not need to access memory or I/O
 - Some instructions act inside processor only, e.g. instruction to move a constant to a register
- When processor accesses memory or an I/O device during execute cycle, there are four possible combinations:

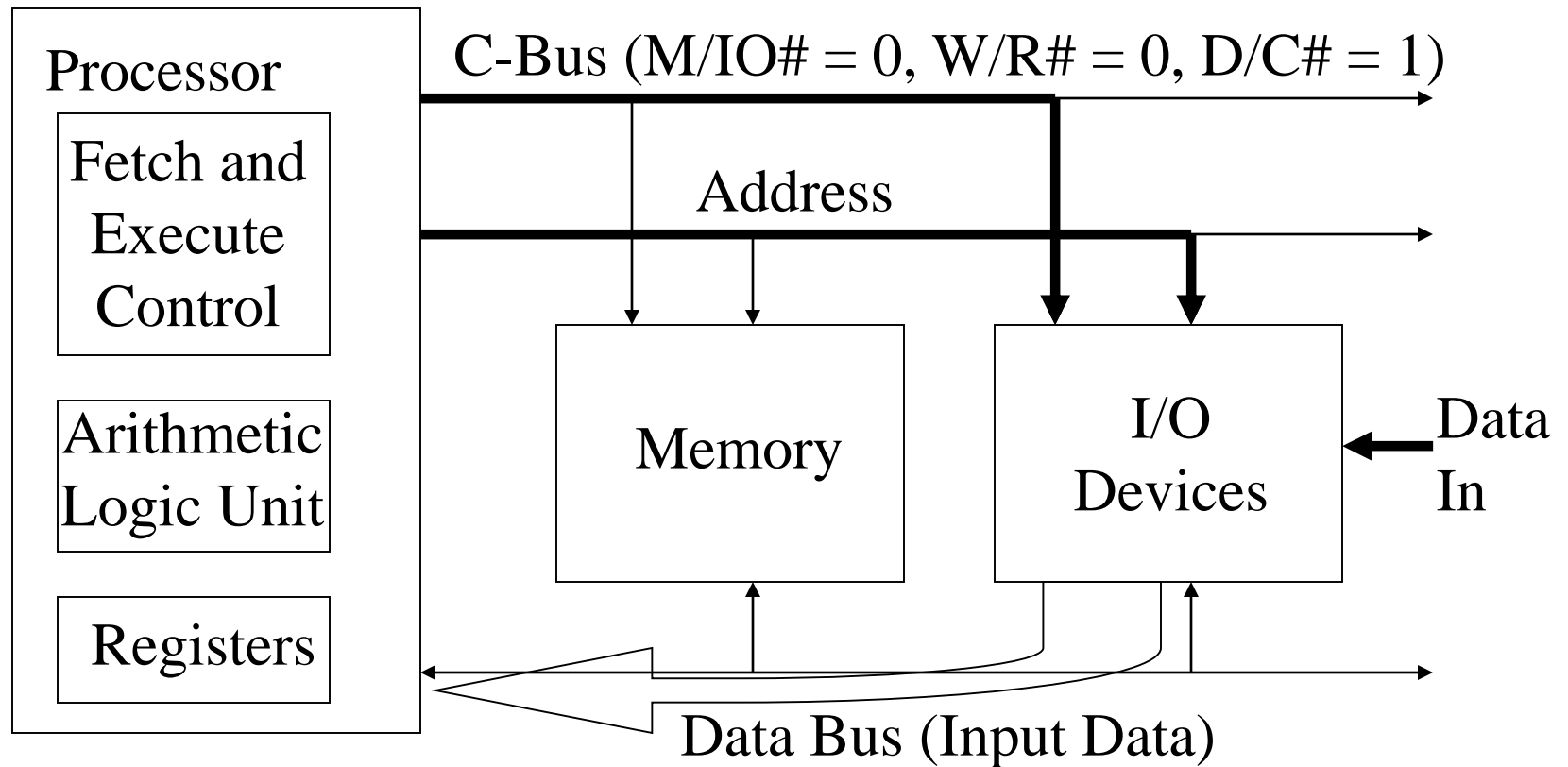
Execute Cycle (Memory Read)



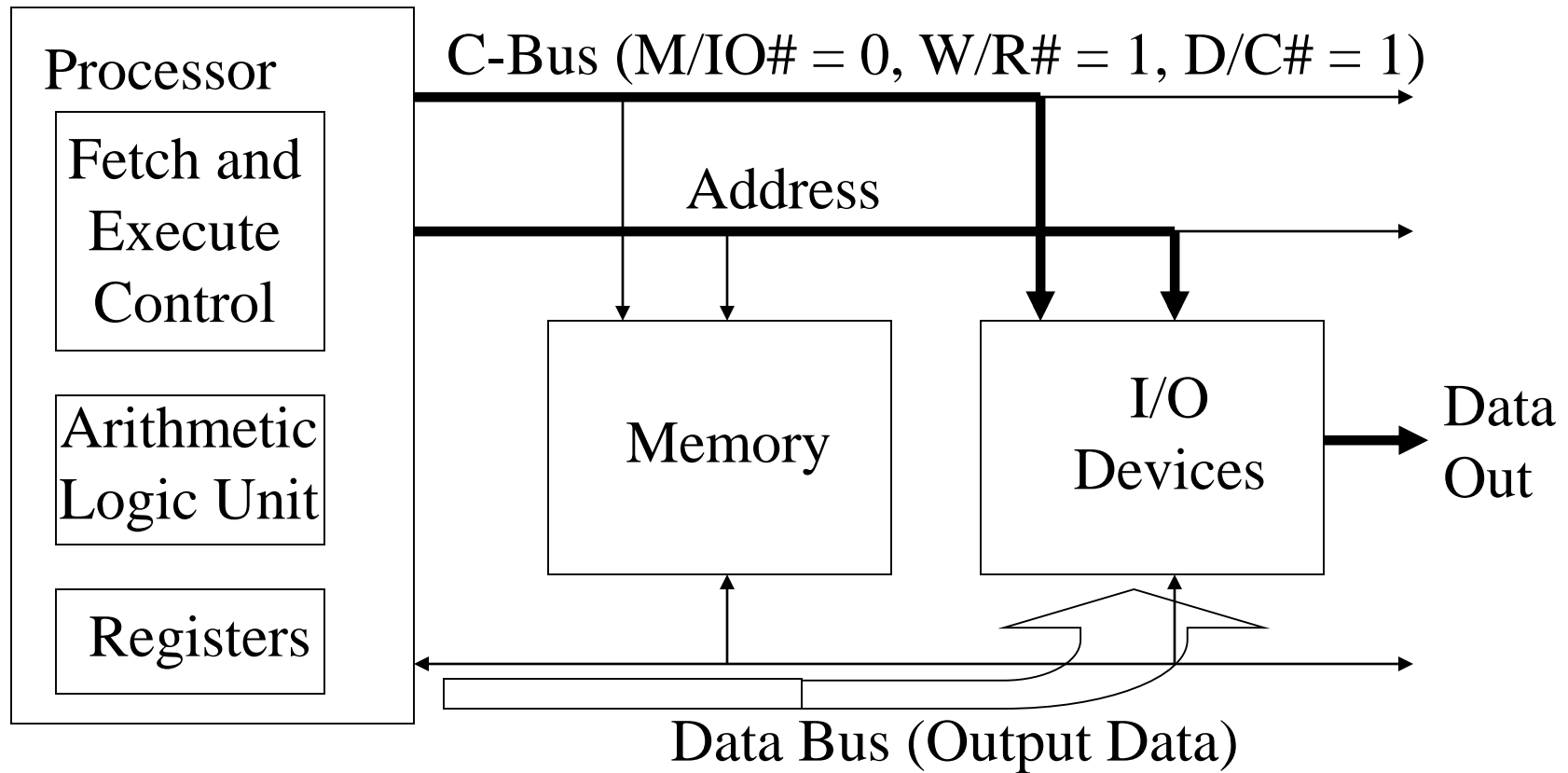
Execute Cycle (Memory Write)



Execute Cycle (I/O Read)



Execute Cycle (I/O Write)



Cache Memory

- Located between CPU and main memory
 - Holds a copy of most recently accessed data
- L1 cache implemented with “hidden” registers
 - Not visible to assembly language programmer
- L2 cache implemented with SRAM memory
 - Faster access than DRAM main memory
 - Typically larger than L1 cache
- Either cache speeds up access to most recently and frequently accessed memory locations
- Avoids a bus cycle to main memory when data is found in cache (“cache hit”)

Cache Memory

