Computer Architecture

Lecture 10

Md. Biplob Hosen
Lecturer, IIT, JU

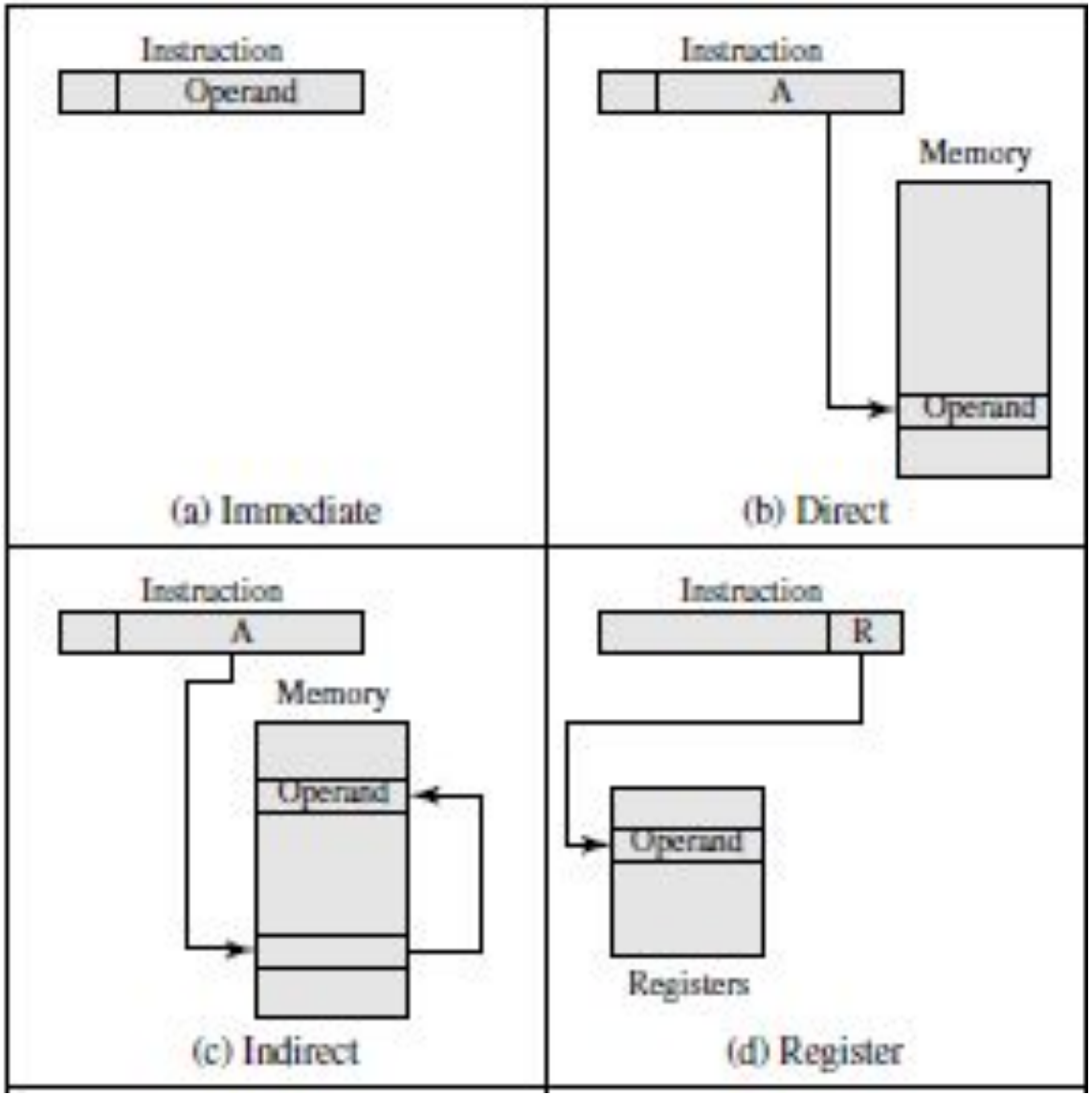
Reference Books

- Computer Organization and Architecture: Designing for Performance- William Stallings (8^{th} Edition)
  - Any later edition is fine

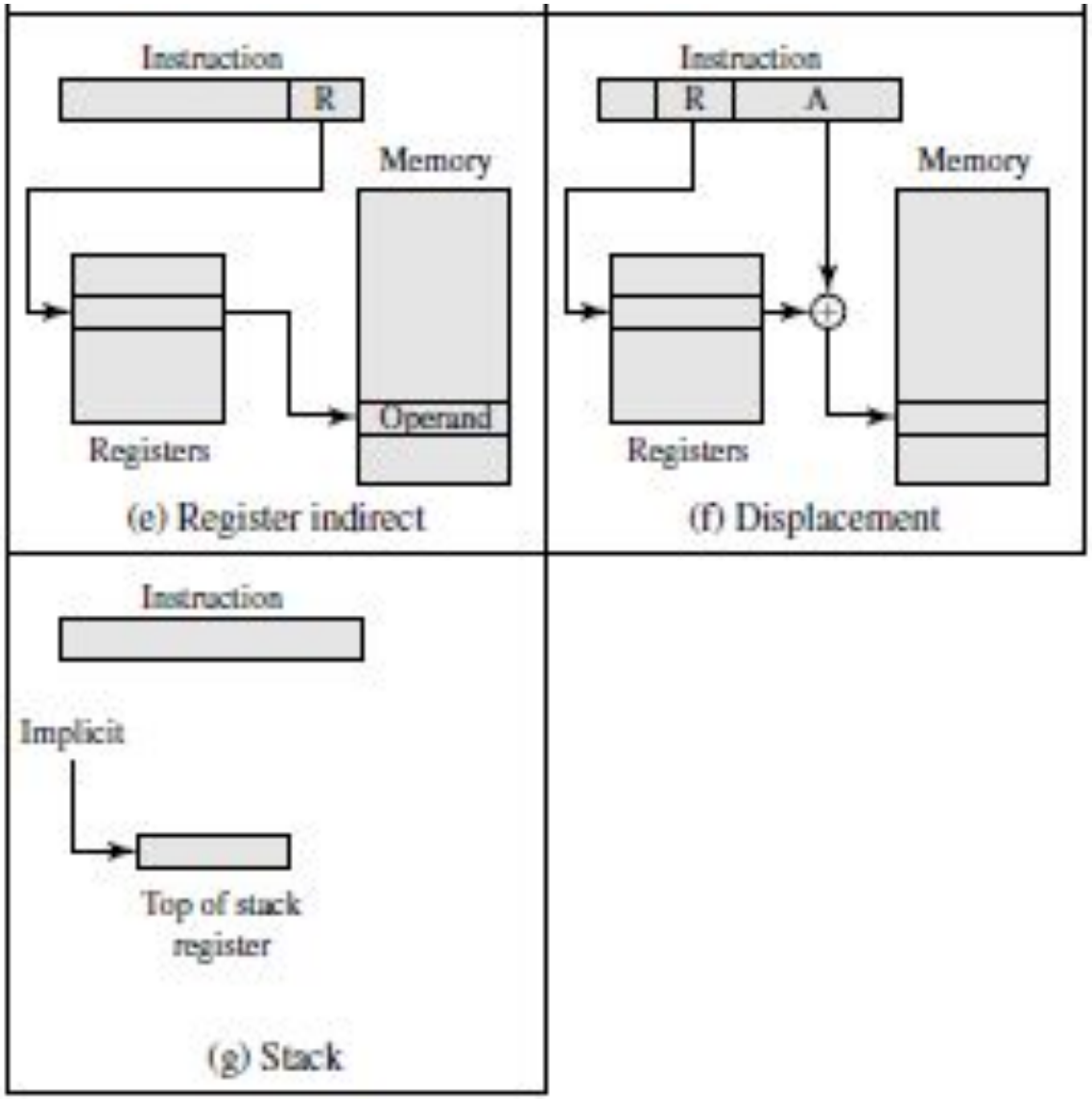
Addressing Techniques

- Depends on trade-off between address range and/or addressing flexibility, and the number of memory references in the instruction and/or the complexity of address calculation
  1) Immediate
  2) Direct
  3) Indirect
  4) Register
  5) Register indirect
  6) Displacement
  7) Stack

Addressing Techniques



Addressing Techniques



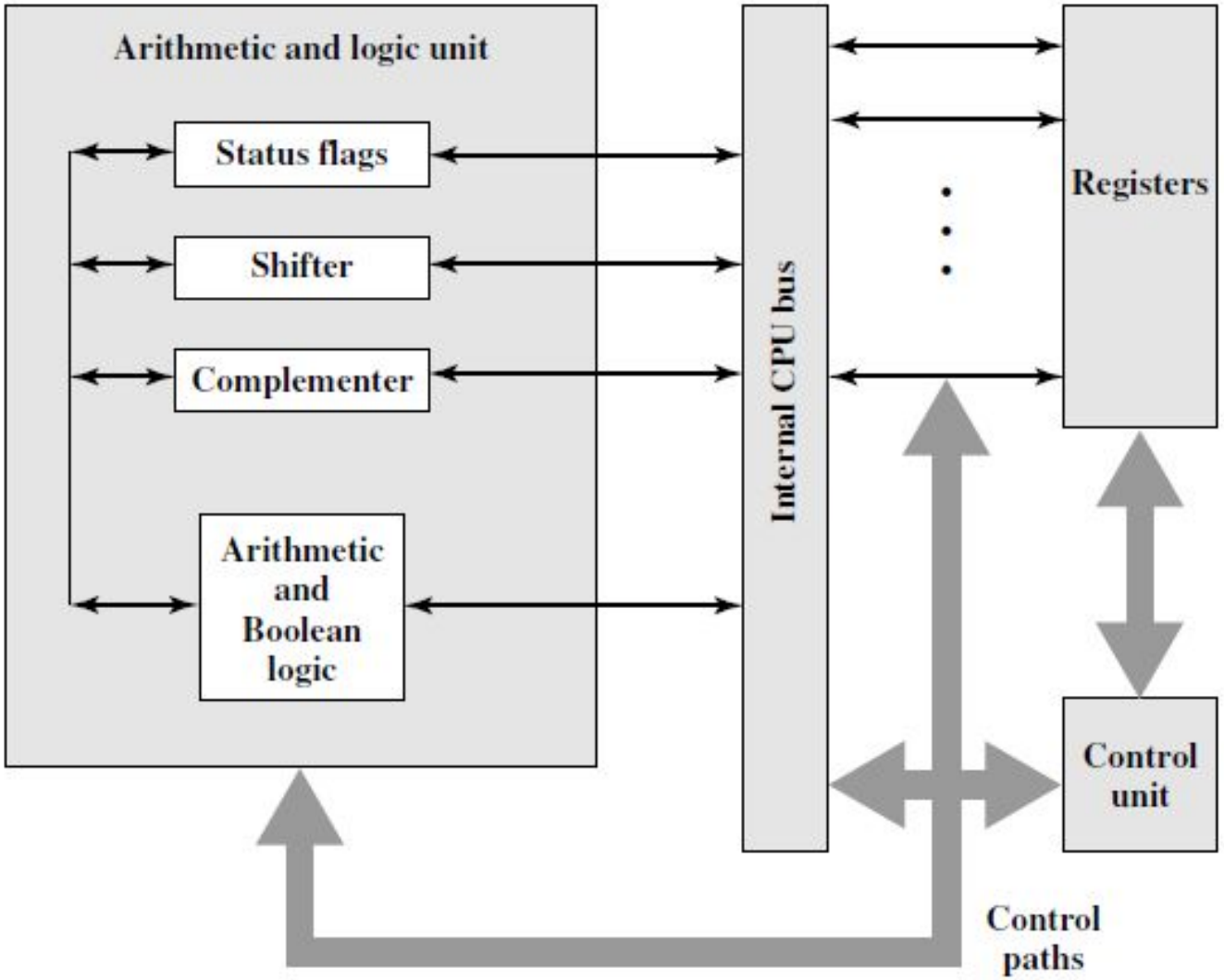
Addressing Techniques

Mode	Algorithm	Principal Advantage	Principal Disadvantage
Immediate	Operand = A	No memory reference	Limited operand magnitude
Direct	EA = A	Simple	Limited address space
Indirect	EA = (A)	Large address space	Multiple memory references
Register	EA = R	No memory reference	Limited address space
Register indirect	EA = (R)	Large address space	Extra memory reference
Displacement	EA = A + (R)	Flexibility	Complexity
Stack	EA = top of stack	No memory reference	Limited applicability

Addressing Techniques

- Virtually all computer architectures provide more than one of these addressing modes
  - Often, different opcodes will use different addressing modes
  - Also, one or more bits in the instruction format can be used as a *mode field*
- Effective address (EA): In a system without virtual memory, the *effective address* will be either a main memory address or a register
- In a virtual memory system, the effective address is a virtual address or a register
- The actual mapping to a physical address is a function of the memory management unit (MMU) and is invisible to the programmer

Internal Structure of the CPU



Register Organization

- The registers in the processor perform two roles:
  - User-visible registers: Enable the machine-or assembly language programmer to minimize main memory references by optimizing use of registers
  - Control and status registers: Used by the control unit to control the operation of the processor and by privileged, operating system programs to control the execution of programs

User-Visible Registers

- Could be categorized into following categories:
  - General purpose
  - Data
  - Address
  - Condition codes
- GP registers can be assigned to a variety of functions by the programmer
  - Any general-purpose register can contain the operand for any opcode
  - Can be used for addressing functions (e.g., register indirect, displacement)

User-Visible Registers

- **Data registers** may be used only to hold data and cannot be employed in the calculation of an operand address
- **Address registers** may themselves be somewhat general purpose, or they may be devoted to a particular addressing mode
- Such as:
  - **Segment pointers:** Holds the address of the base of the segment.
  - **Index registers:** These are used for indexed addressing and may be auto indexed
  - **Stack pointer:** If there is user-visible stack addressing, then typically there is a dedicated register that points to the top of the stack
  - This allows implicit addressing; that is, push, pop, and other stack instructions need not contain an explicit stack operand

User-Visible Registers

- A partially visible registers, which hold condition codes (also referred to as *flags*)
- Condition codes are bits set by the processor hardware as the result of operations
- Example: An arithmetic operation may produce a positive, negative, zero, or overflow result
- In addition to the result itself being stored in a register or memory, a condition code is also set
- The code may subsequently be tested as part of a conditional branch operation

Condition Codes

Advantages	Disadvantages
1. Because condition codes are set by normal arithmetic and data movement instructions, they should reduce the number of COMPARE and TEST instructions needed. 2. Conditional instructions, such as BRANCH are simplified relative to composite instructions, such as TEST AND BRANCH. 3. Condition codes facilitate multiway branches. For example, a TEST instruction can be followed by two branches, one on less than or equal to zero and one on greater than zero.	1. Condition codes add complexity, both to the hardware and software. Condition code bits are often modified in different ways by different instructions, making life more difficult for both the microprogrammer and compiler writer. 2. Condition codes are irregular; they are typically not part of the main data path, so they require extra hardware connections. 3. Often condition code machines must add special non-condition-code instructions for special situations anyway, such as bit checking, loop control, and atomic semaphore operations. 4. In a pipelined implementation, condition codes require special synchronization to avoid conflicts.

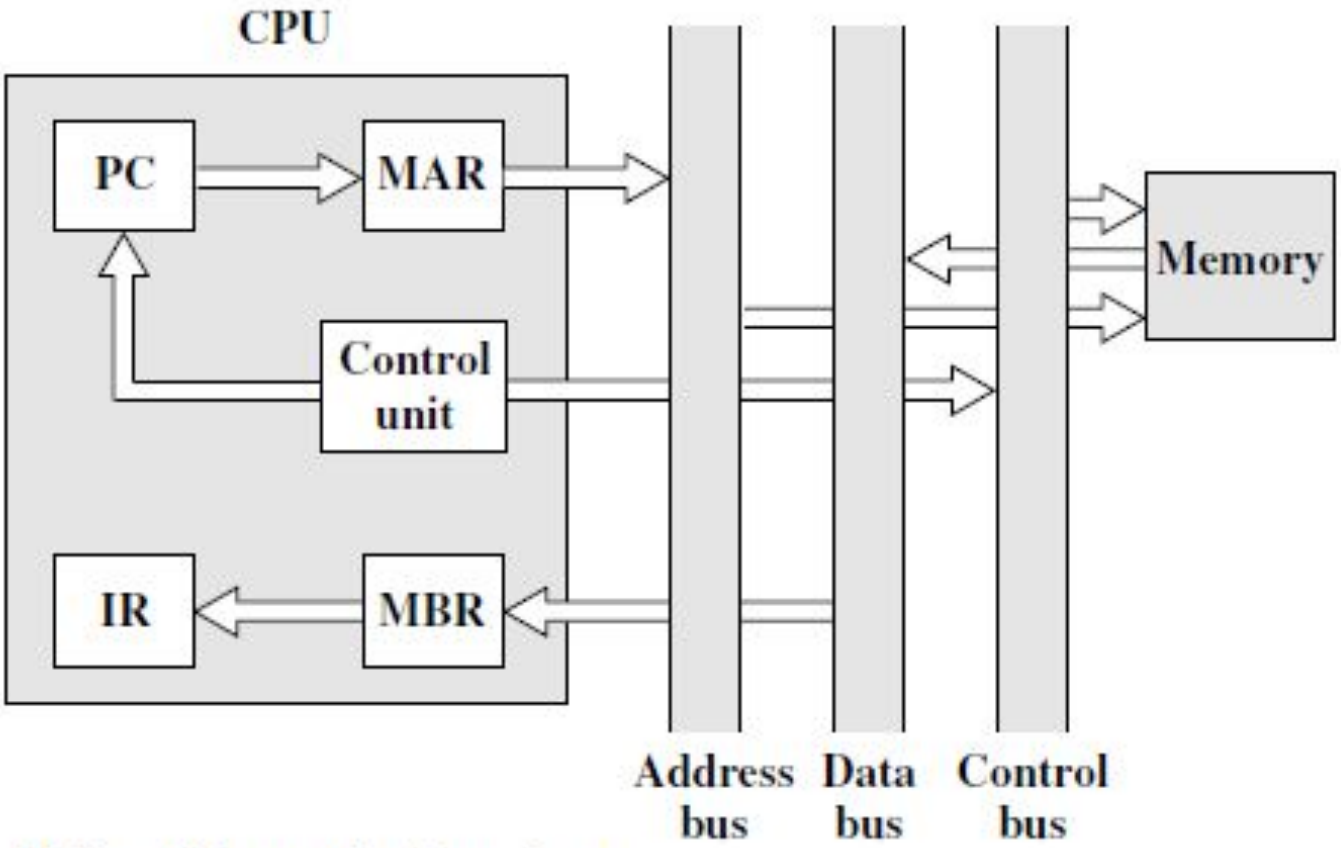
Control and Status Registers

- Four main categories:
- Program counter (PC): Contains the address of an instruction to be fetched
- Instruction register (IR): Contains the instruction most recently fetched
- Memory address register (MAR): Contains the address of a location in memory
- Memory buffer register (MBR): Contains a word of data to be written to memory or the word most recently read

Control and Status Registers

- Program Status Word (PSW)
- Contains condition codes plus other status information. Common fields or flags include the following:
  - Sign: Contains the sign bit of the result of the last arithmetic operation
  - Zero: Set when the result is 0
  - Carry: Set if an operation resulted in a carry (addition) into or borrow (subtraction) out of a high-order bit
  - Equal: Set if a logical compare result is equality
  - Overflow: Used to indicate arithmetic overflow
  - Interrupt Enable/Disable: Used to enable or disable interrupts
  - Supervisor: Indicates whether the processor is executing in supervisor or user mode

Data Flow



MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

Figure: Data Flow, Fetch Cycle

Data Flow

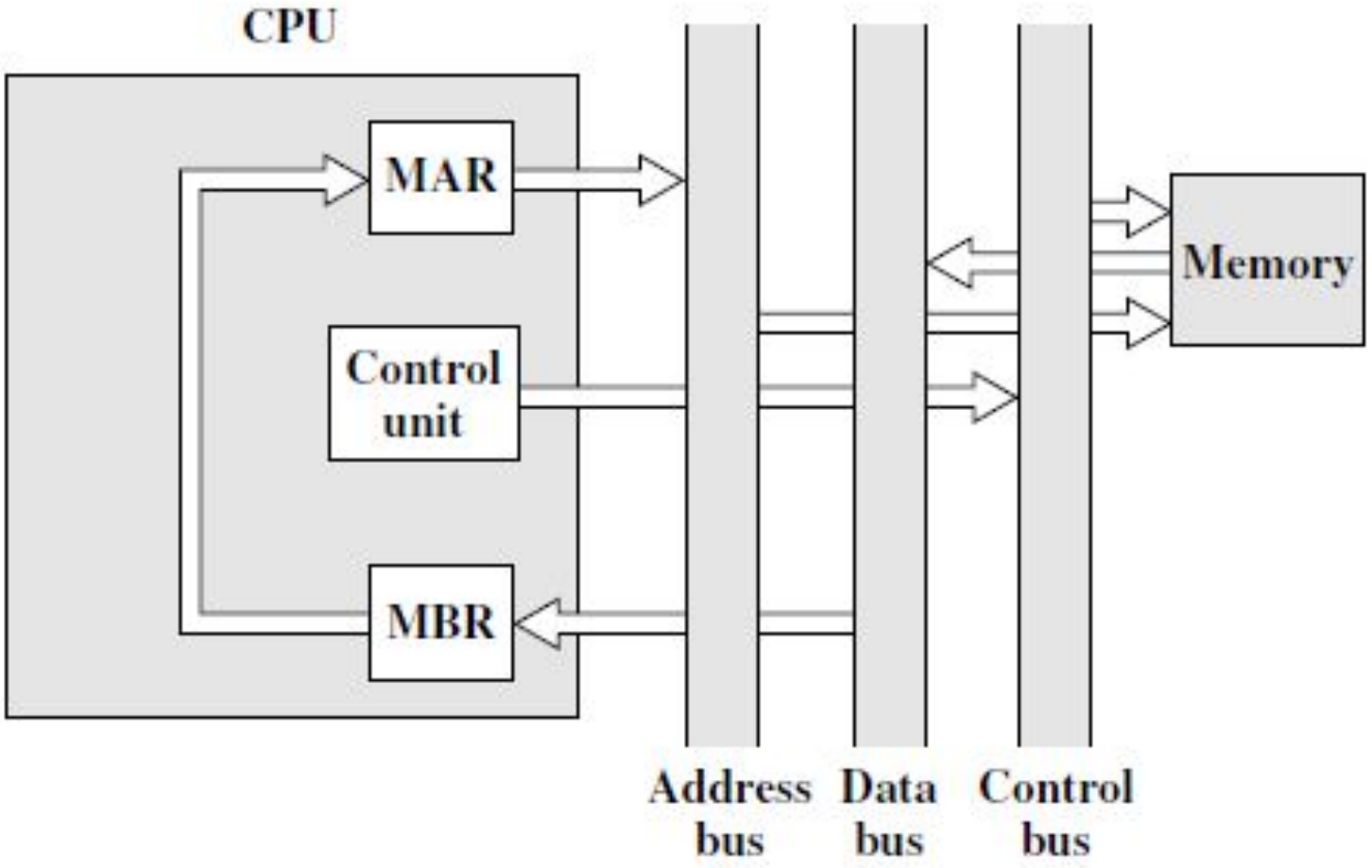


Figure: Data Flow, Indirect Cycle

Data Flow

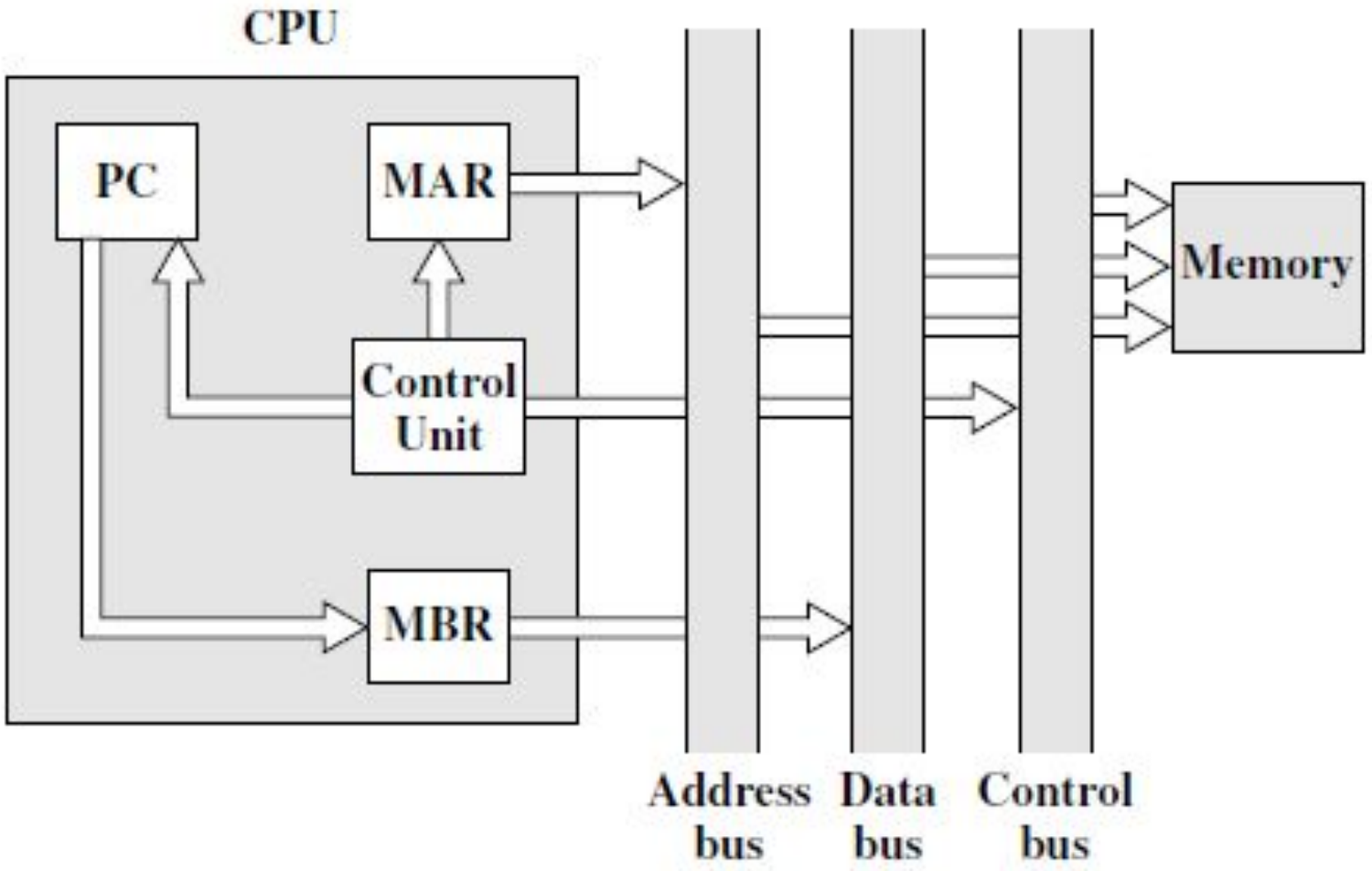


Figure: Data Flow, Interrupt Cycle

Thank you!