UNIT-II

Control Unit:
Control Unit and instruction set,
Hardwired control and microprogrammed control,
Microoperation sequencing logic,
Control memory

+ Instruction set architecture:
Instruction set,
instruction cycle and state diagram,
Instruction factors, addressing modes
CISE and RISE systems

Anshul (PhD)
Assistant Professor
Department of CSE
NIT Patna, Bihar-80005

Computer Components

- Contemporary computer designs are based on concepts developed by John von Neumann at the Institute for Advanced Studies, Princeton
- Referred to as the *von Neumann architecture* and is based on three key concepts:
  - Data and instructions are stored in a single read-write memory
  - The contents of this memory are addressable by location, without regard to the type of data contained there
  - Execution occurs in a sequential fashion (unless explicitly modified) from one instruction to the next
- *Hardwired program*
  - The result of the process of connecting the various components in the desired configuration

Hardware and Software Approaches

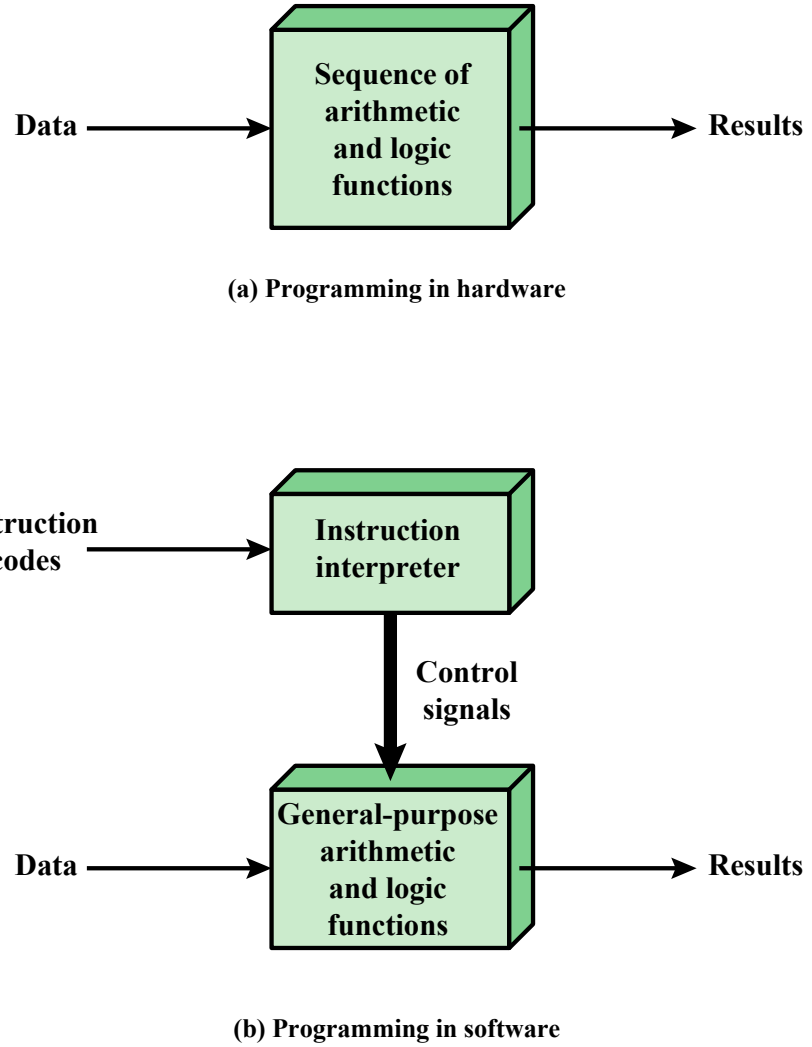


Figure 3.1 Hardware and Software Approaches

Software

- A sequence of codes or instructions
- Part of the hardware interprets each instruction and generates control signals
- Provide a new sequence of codes for each new program instead of rewiring the hardware

Major components:

- CPU
  - Instruction interpreter
  - Module of general-purpose arithmetic and logic functions
- I/O Components
  - Input module
    - Contains basic components for accepting data and instructions and converting them into an internal form of signals usable by the system
  - Output module
    - Means of reporting results

Software

I/O Components

Memory address register (MAR)

- Specifies the address in memory for the next read or write

Memory buffer register (MBR)

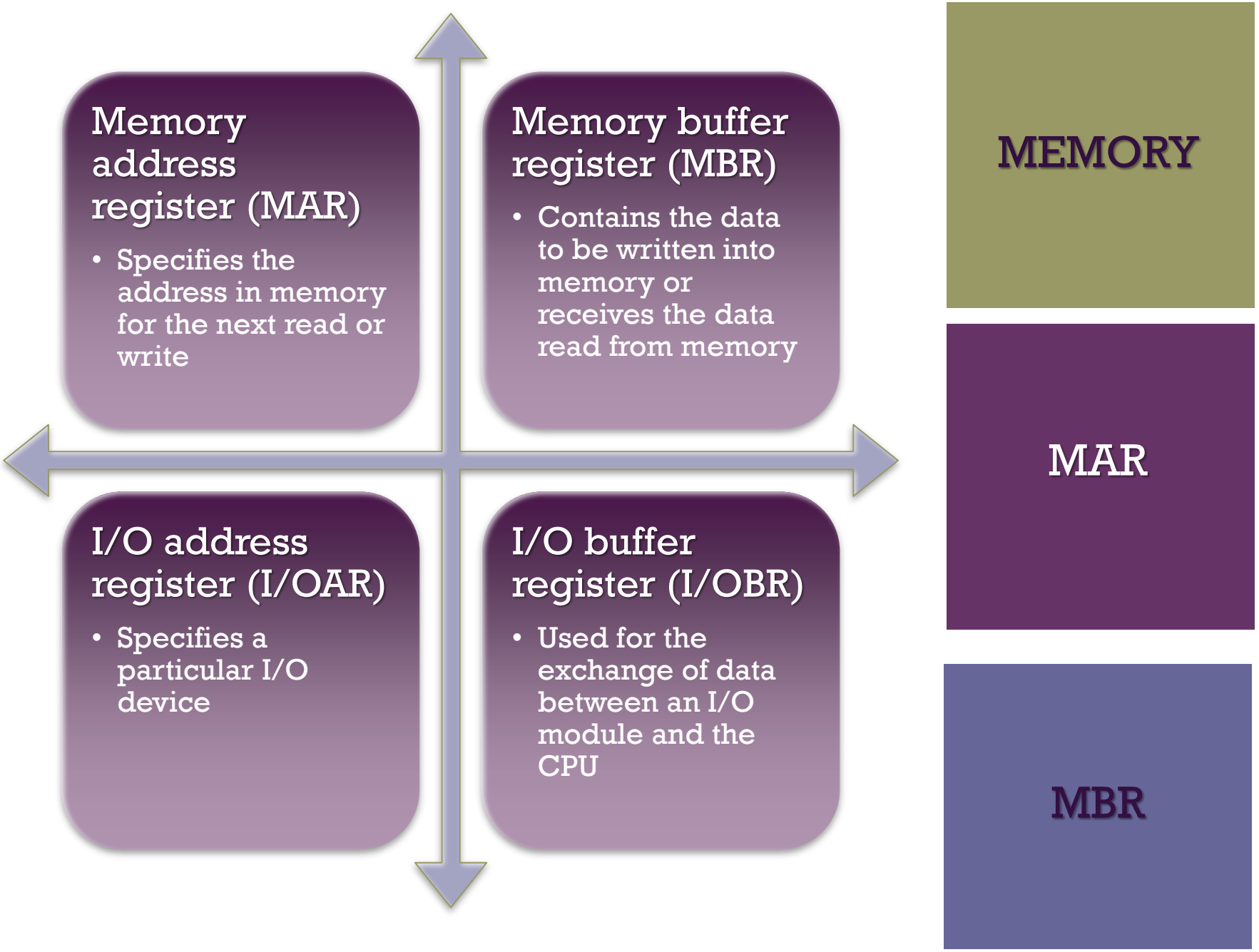
- Contains the data to be written into memory or receives the data read from memory

I/O address register (I/OAR)

- Specifies a particular I/O device

I/O buffer register (I/OBR)

- Used for the exchange of data between an I/O module and the CPU



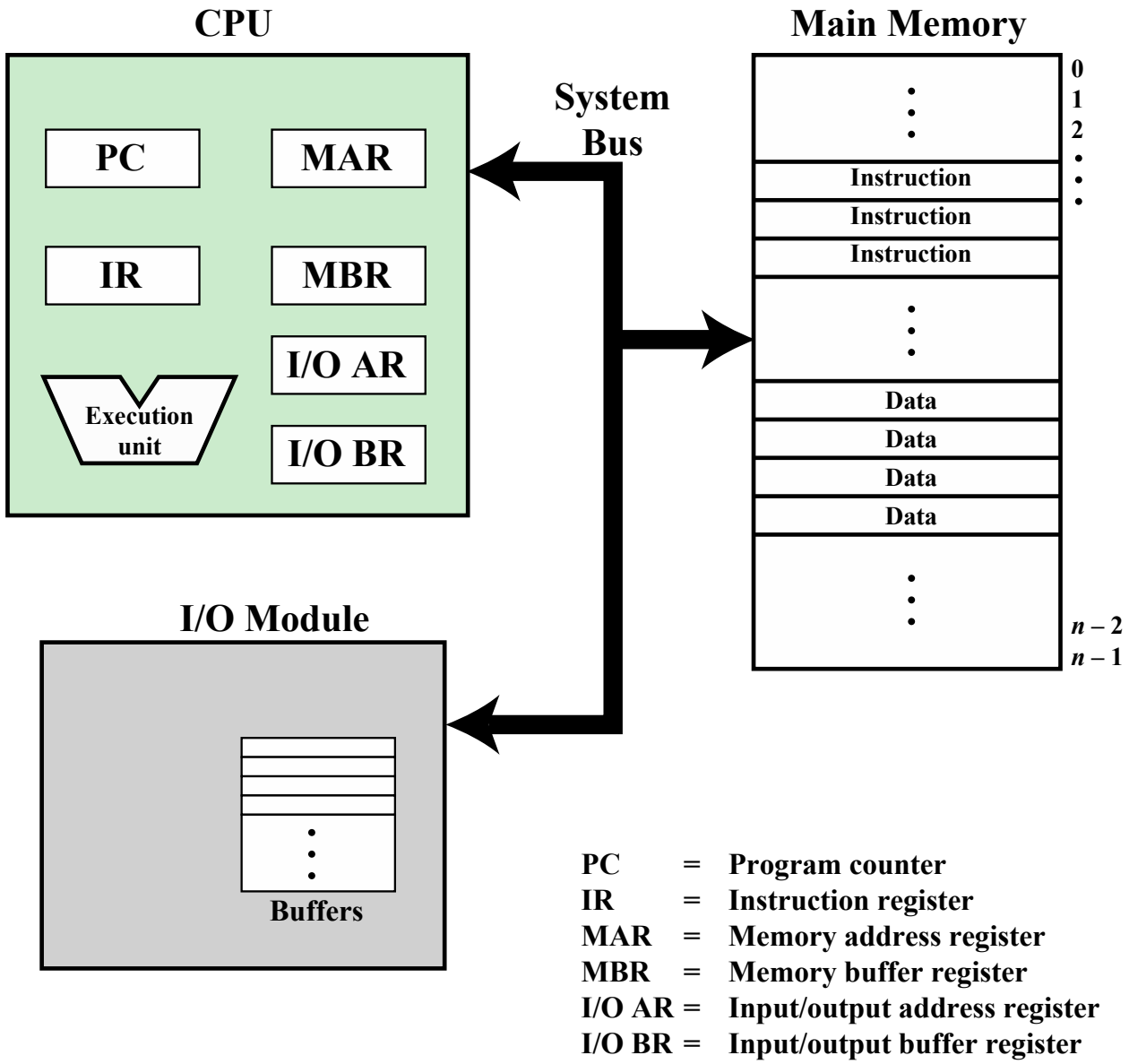


Figure 3.2 Computer Components: Top-Level View

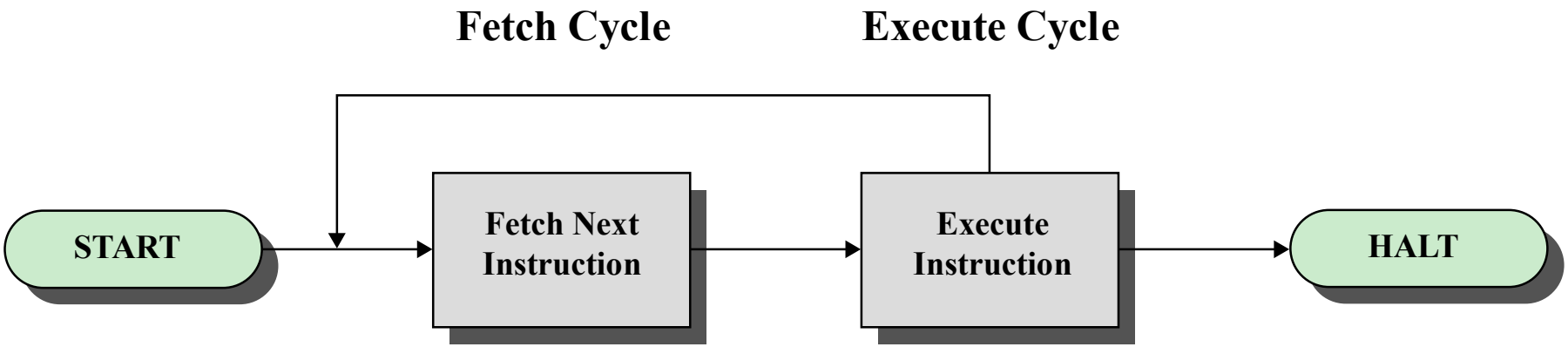


Figure 3.3 Basic Instruction Cycle

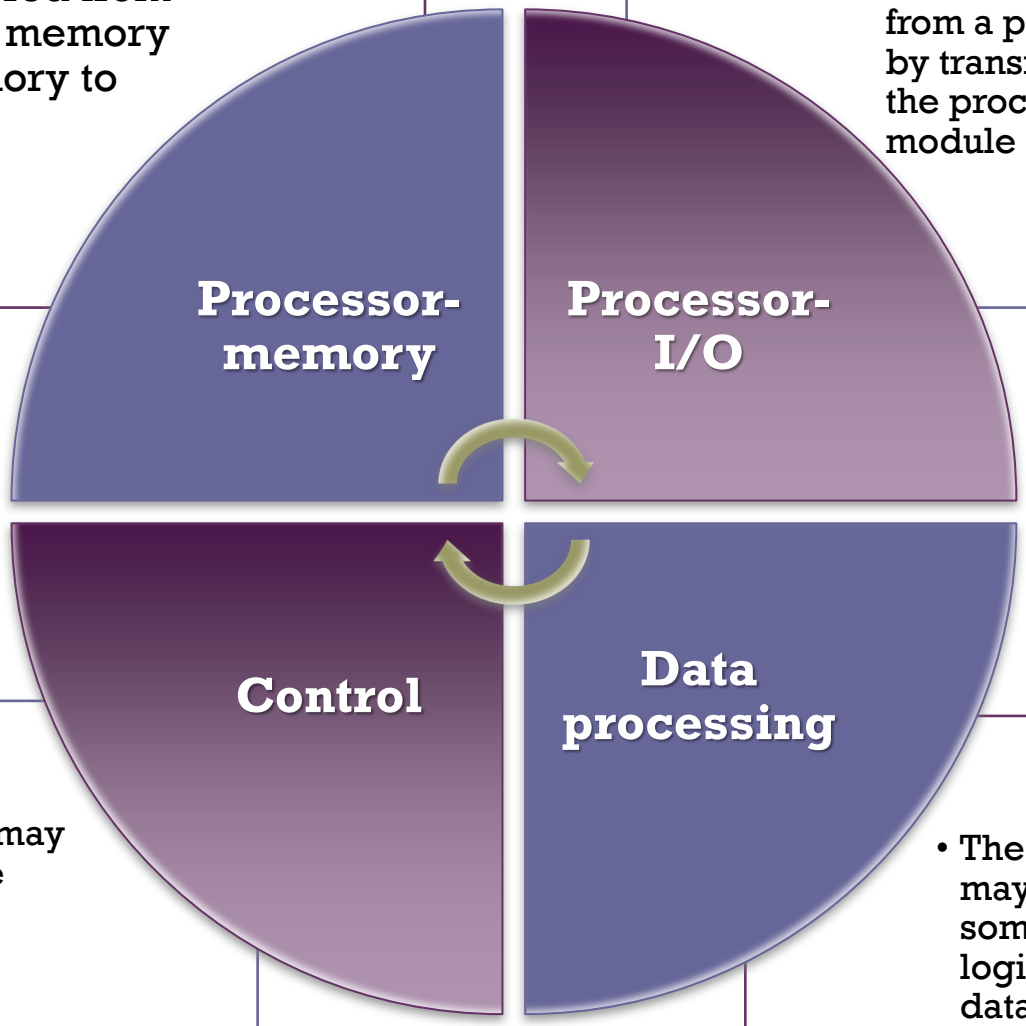
Fetch Cycle

- At the beginning of each instruction cycle the processor fetches an instruction from memory
- The program counter (PC) holds the address of the instruction to be fetched next
- The processor increments the PC after each instruction fetch so that it will fetch the next instruction in sequence
- The fetched instruction is loaded into the instruction register (IR)
- The processor interprets the instruction and performs the required action

Action Categories

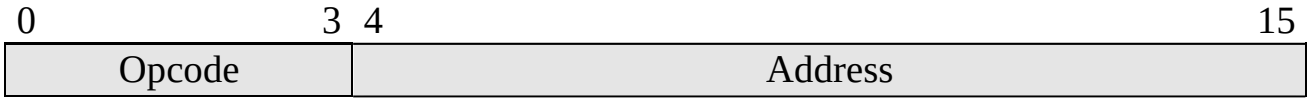
- Data transferred from processor to memory or from memory to processor

- Data transferred to or from a peripheral device by transferring between the processor and an I/O module

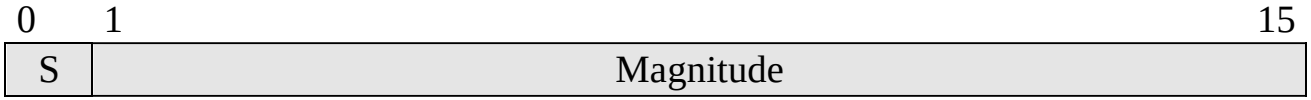


- An instruction may specify that the sequence of execution be altered

- The processor may perform some arithmetic or logic operation on data



(a) Instruction format



(b) Integer format

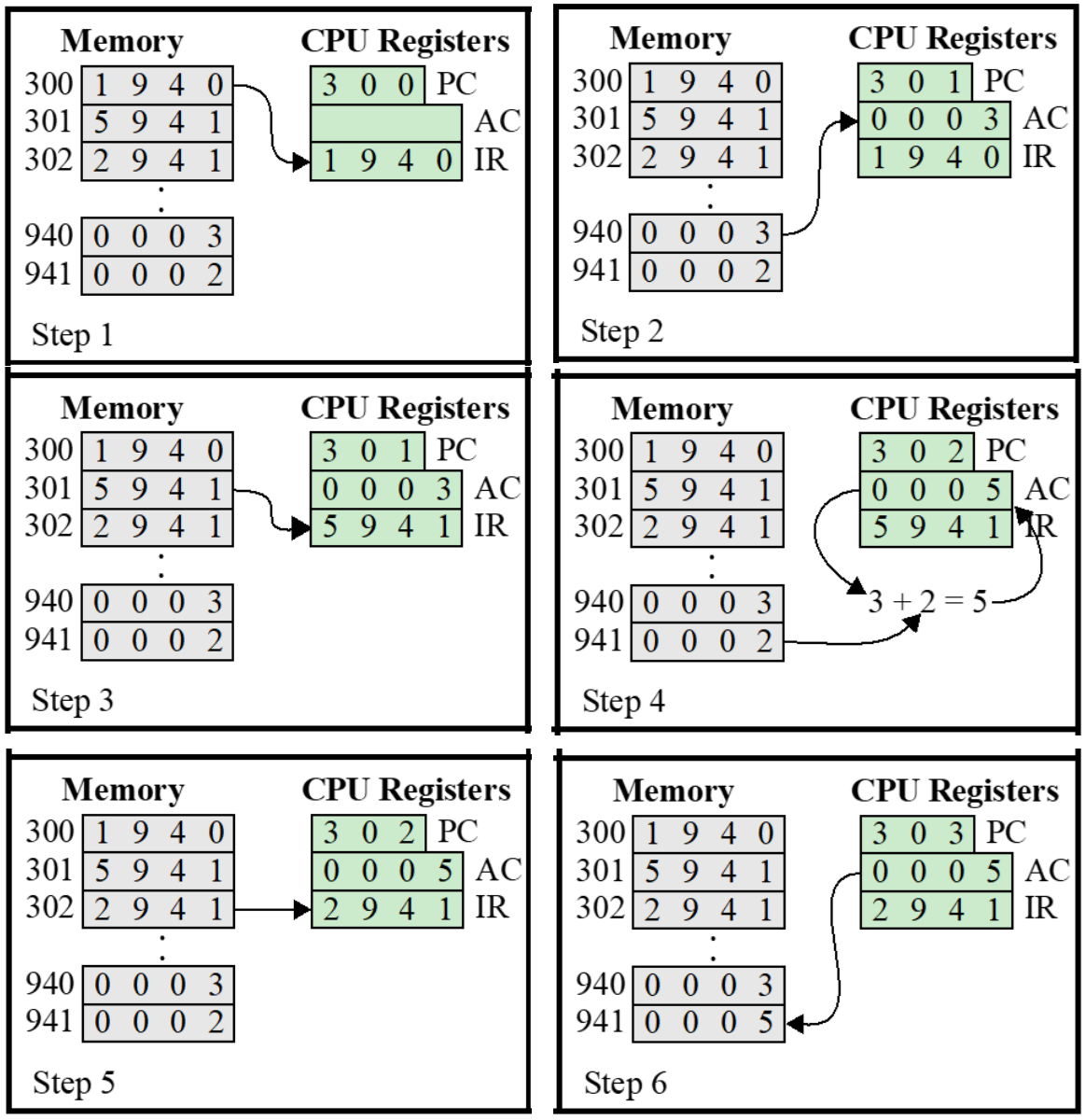
Program Counter (PC) = Address of instruction
Instruction Register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from Memory
0010 = Store AC to Memory
0101 = Add to AC from Memory

(d) Partial list of opcodes

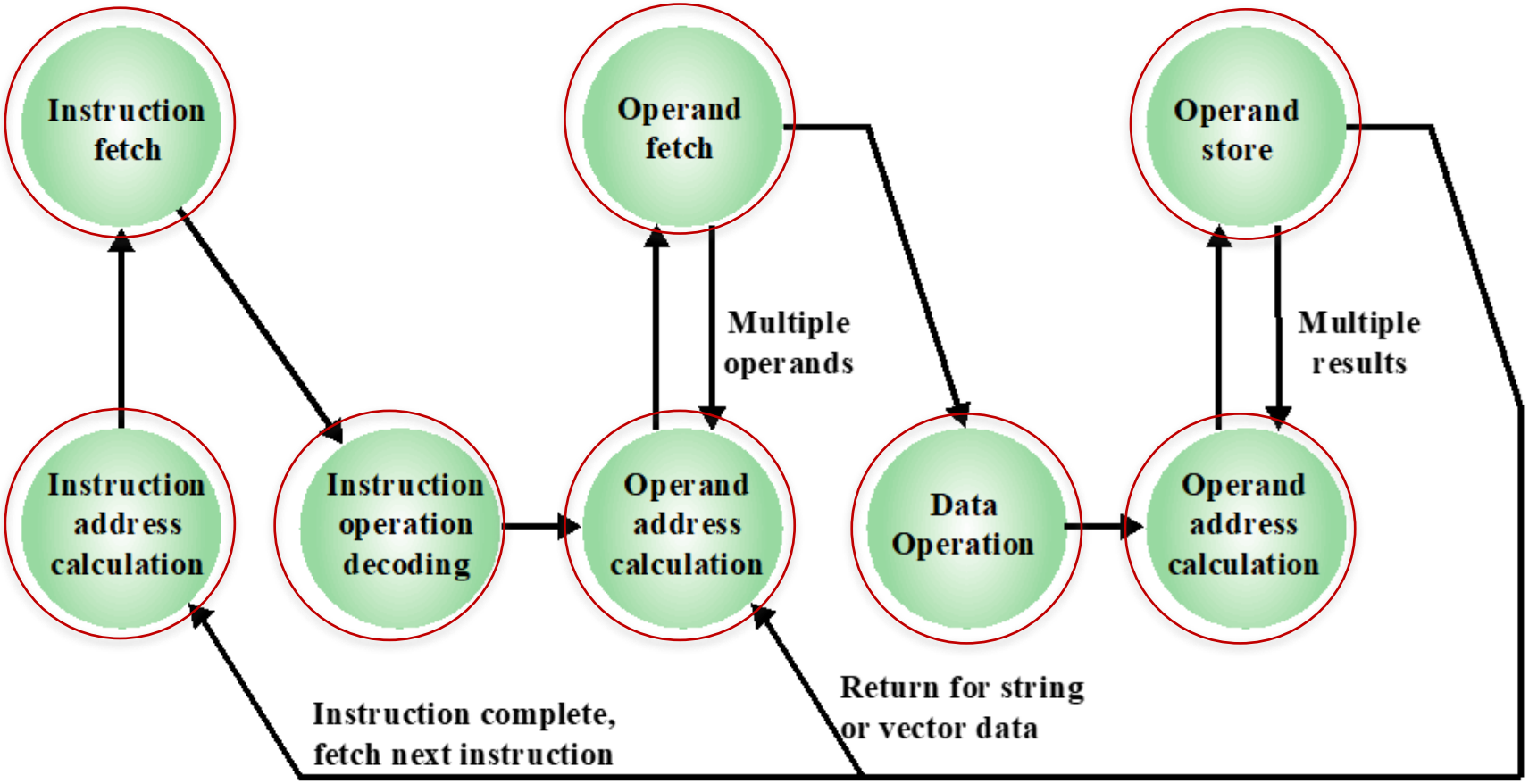
Figure 3.4 Characteristics of a Hypothetical Machine



Instruction cycle

- Instruction address calculation
- Instruction fetch
- Instruction operation decoding
- Operand address calculation
- Operand fetch
- Data operation
- Operand store

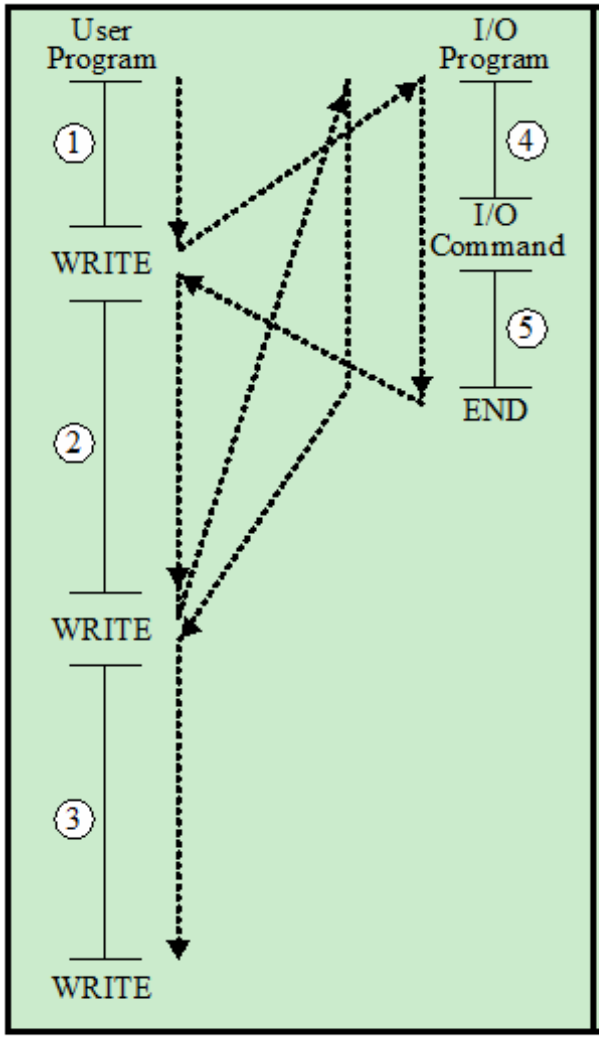
Instruction Cycle Diagram



Classes of Interrupts

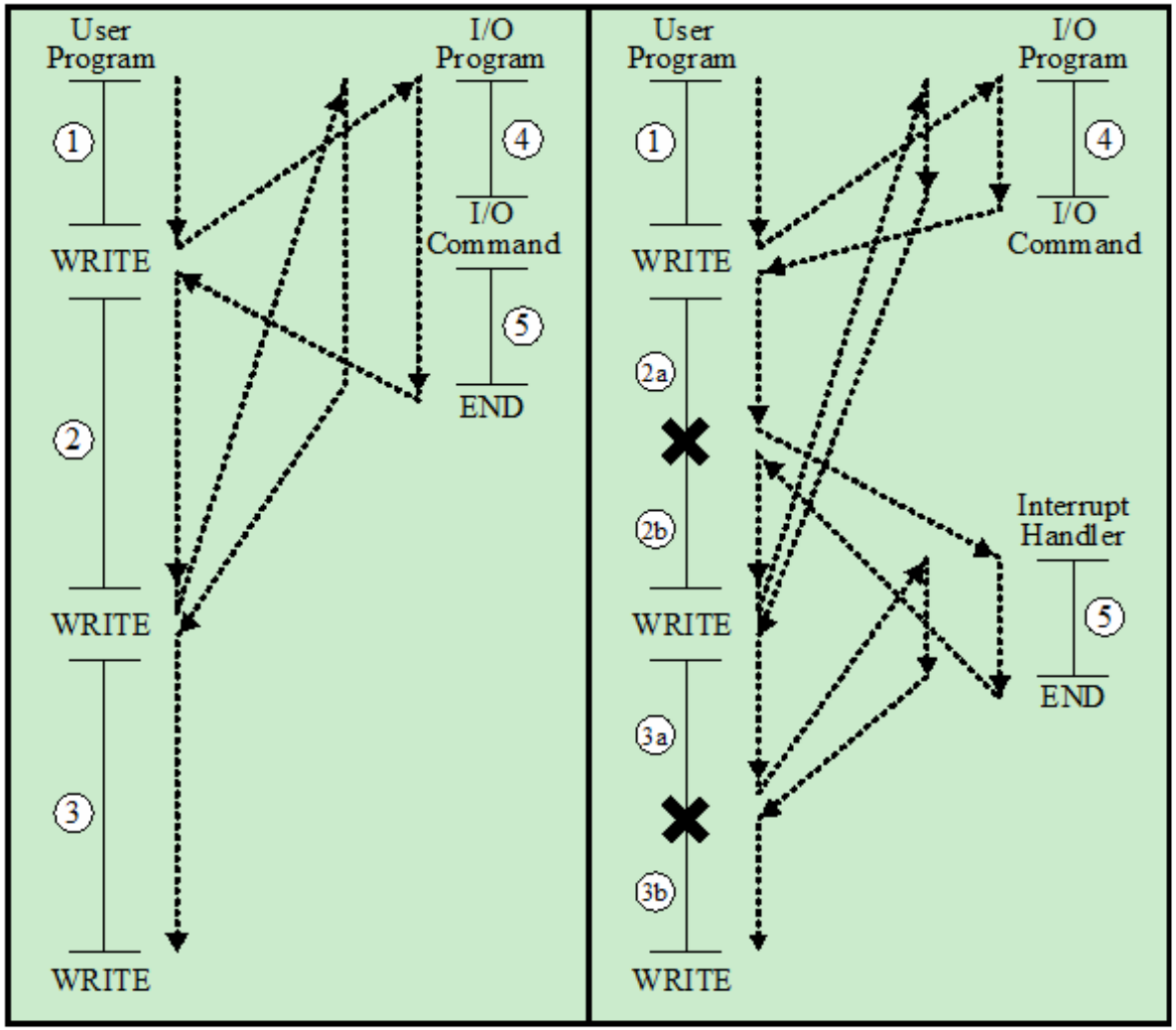
Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, or reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation, request service from the processor, or to signal a variety of error conditions.
Hardware failure	Generated by a failure such as power failure or memory parity error.

Program flow of control



(a) No interrupts

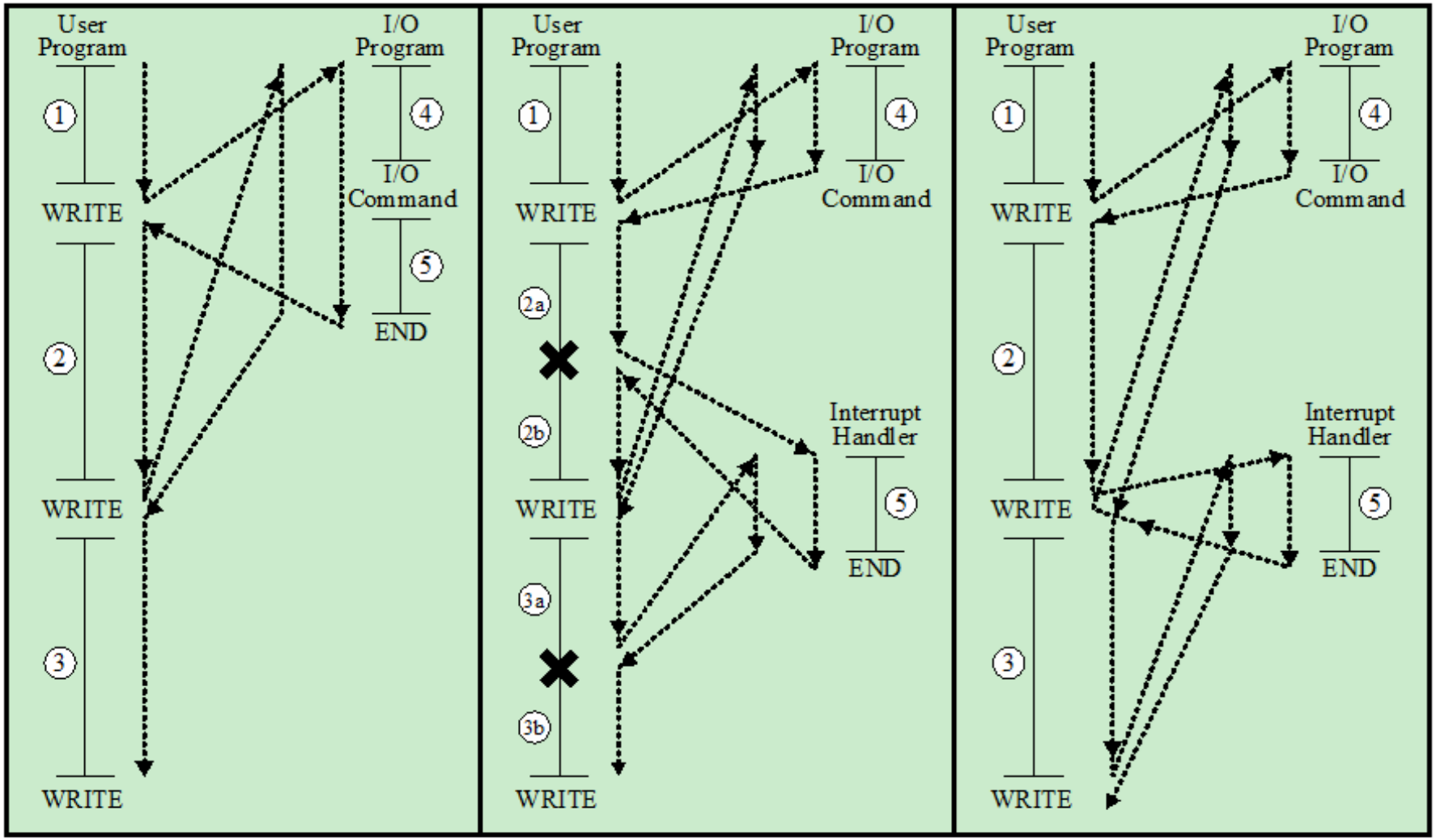
Program flow of control



(a) No interrupts

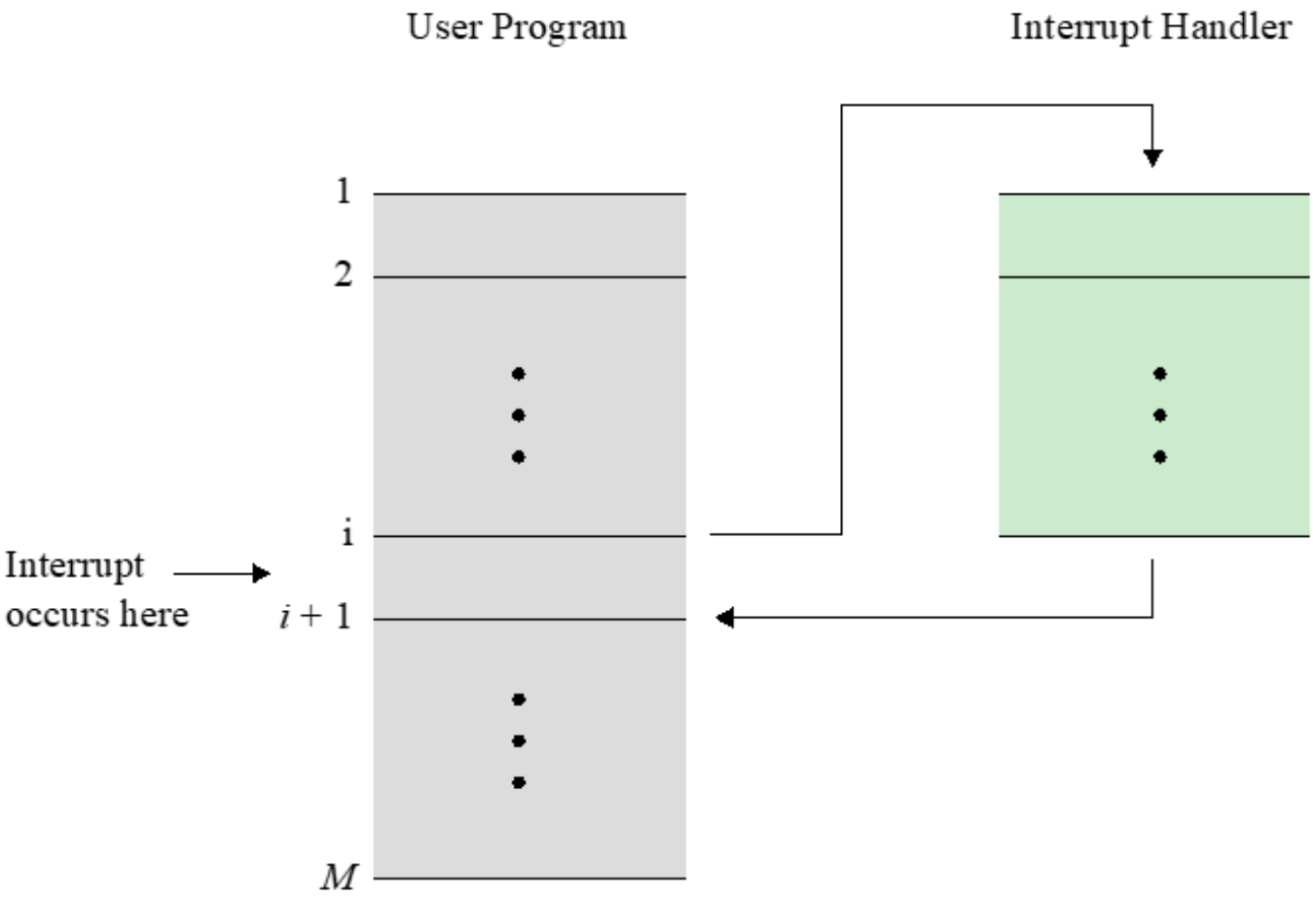
(b) Interrupts; short I/O wait

Program flow of control

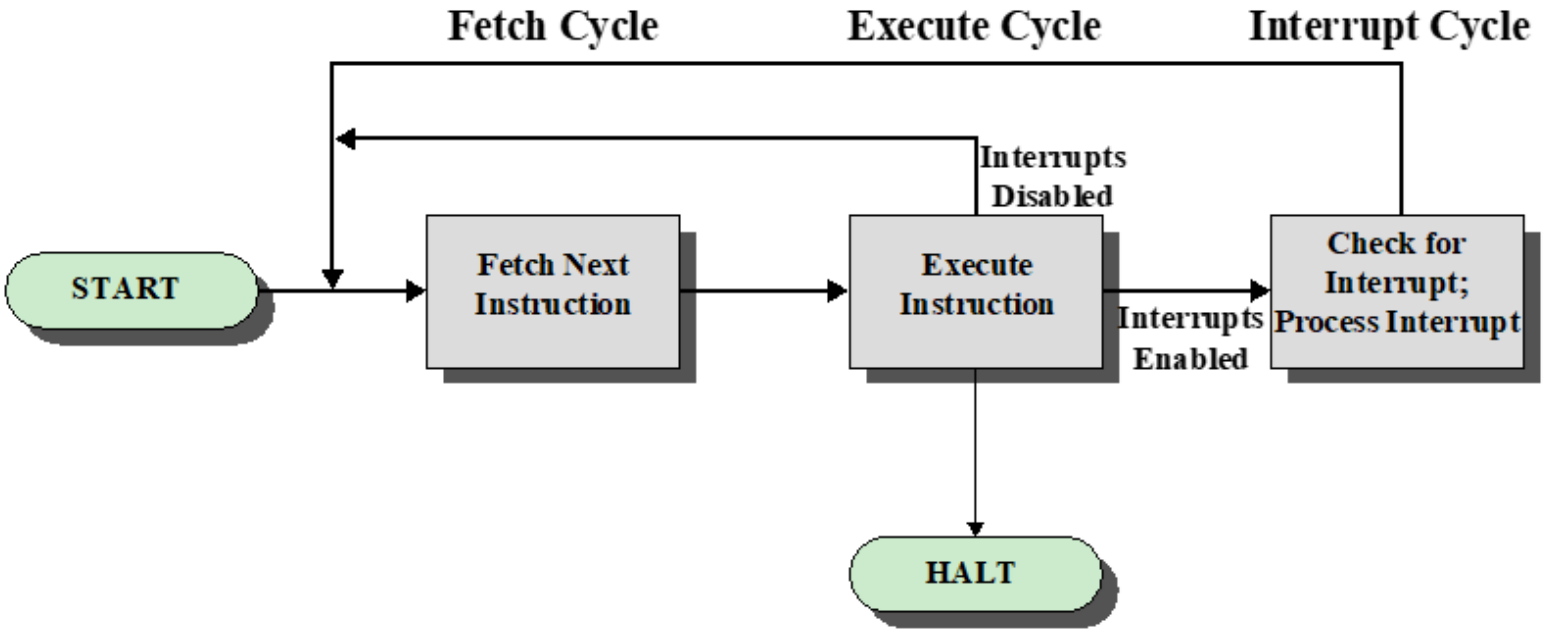


(a) No interrupts (b) Interrupts; short I/O wait (c) Interrupts; long I/O wait

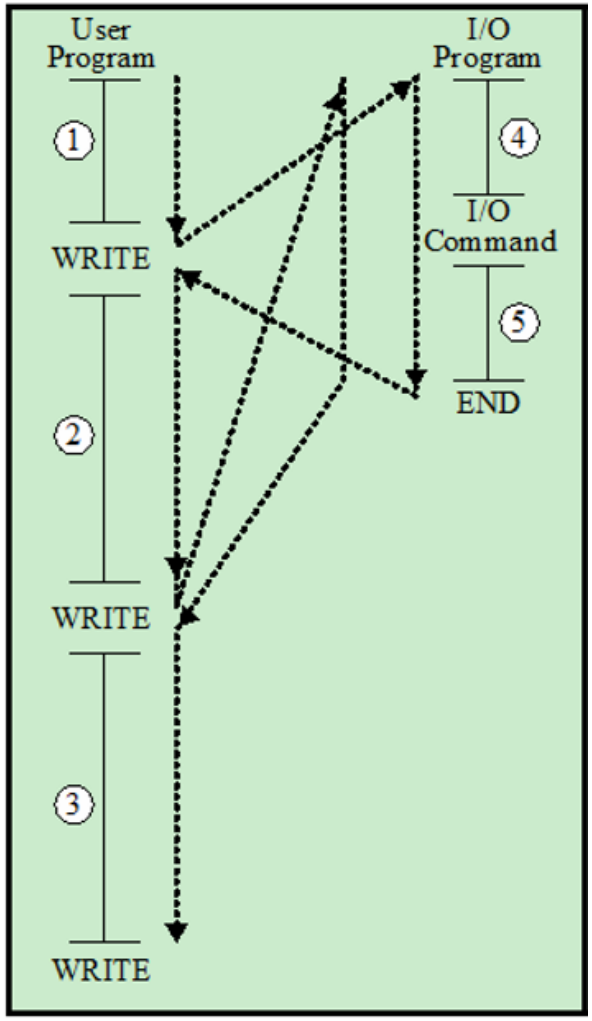
Transfer of Control via Interrupts



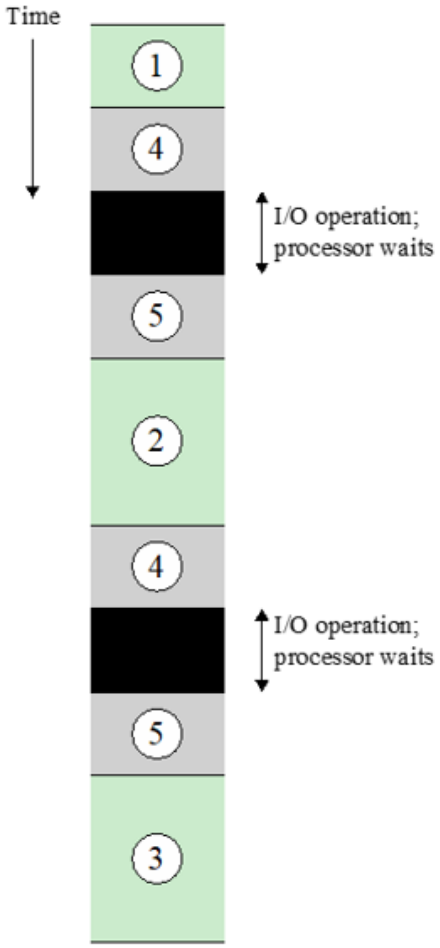
Instruction Cycle with Interrupts



Program Timing: without Interrupt

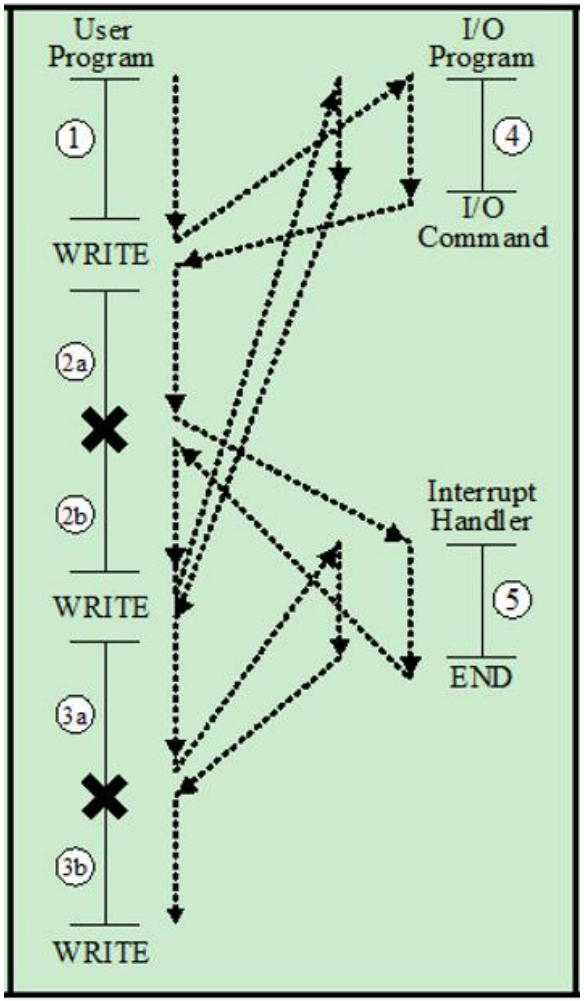


(a) No interrupts

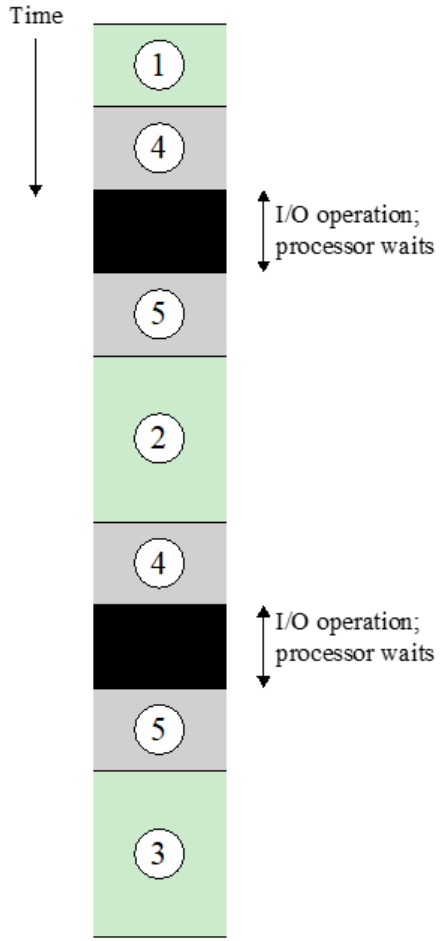


(a) Without interrupts

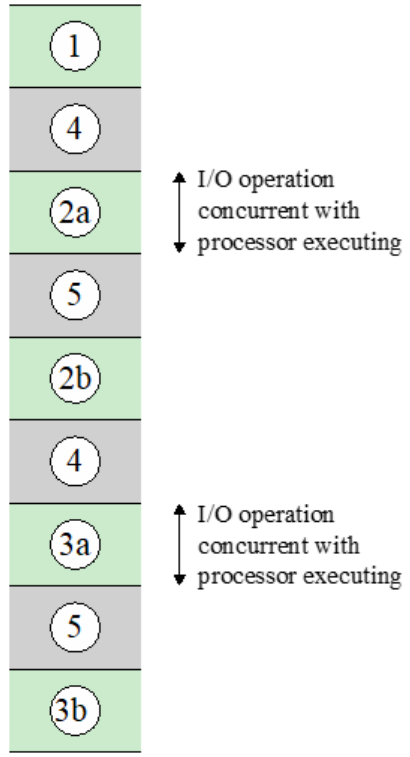
Program Timing: short I/O wait



(b) Interrupts; short I/O wait

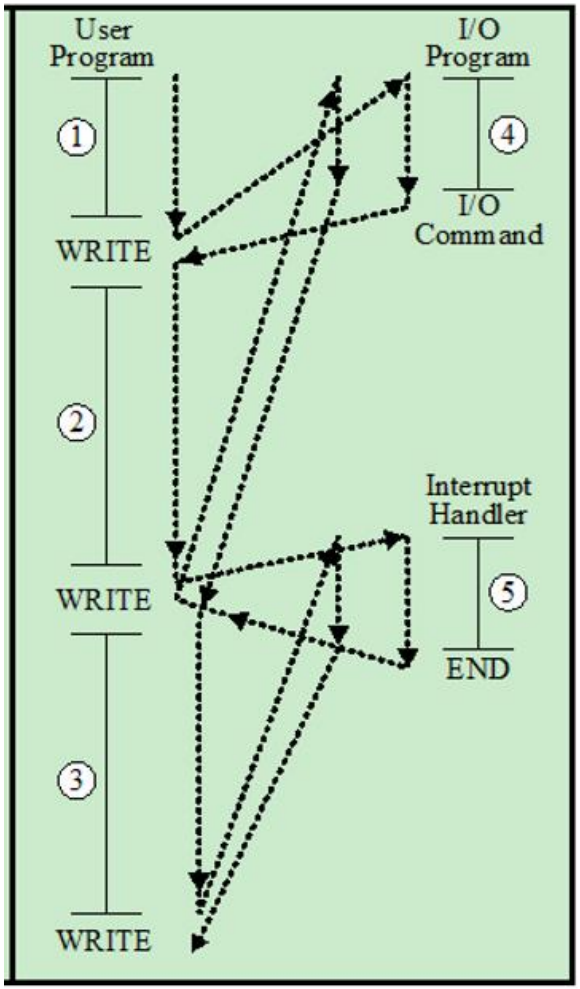


(a) Without interrupts

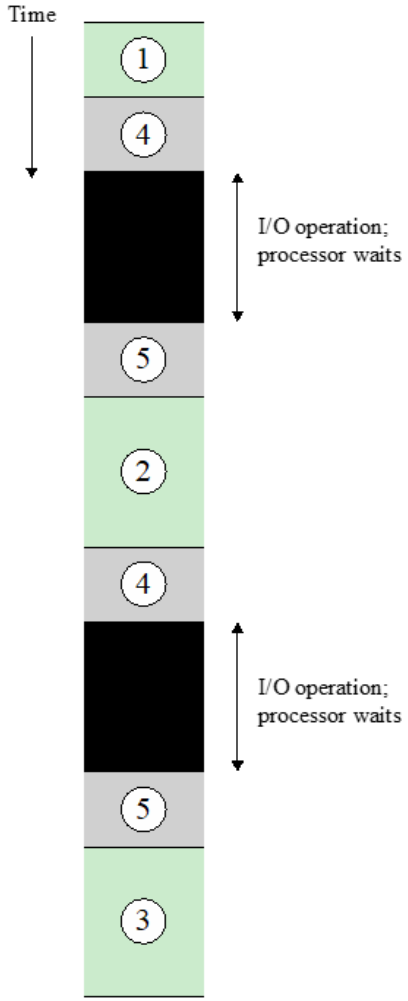


(b) With interrupts

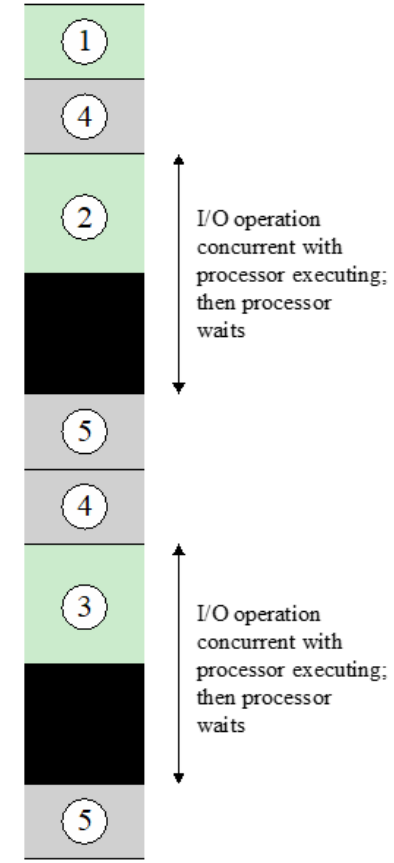
Program Timing: long I/O wait



(c) Interrupts; long I/O wait



(a) Without interrupts

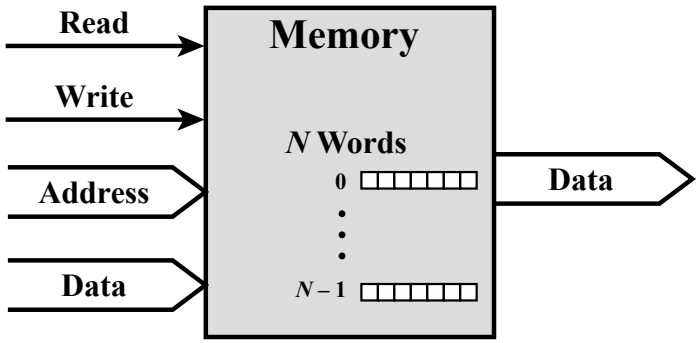


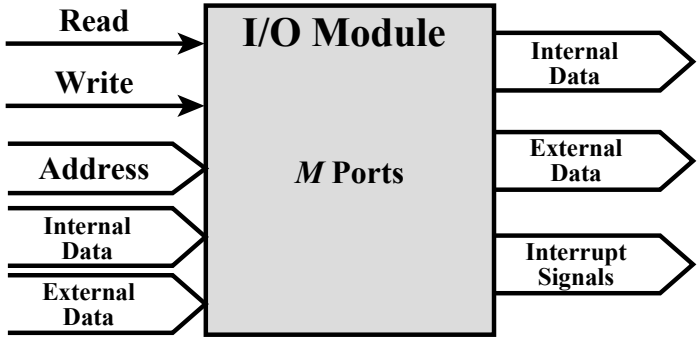
(b) With interrupts

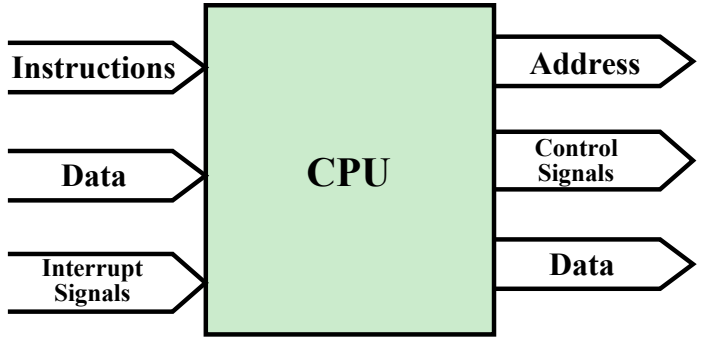
I/O Function

- I/O module can exchange data directly with the processor
- Processor can read data from or write data to an I/O module
  - Processor identifies a specific device that is controlled by a particular I/O module
  - I/O instructions rather than memory referencing instructions
- In some cases, it is desirable to allow I/O exchanges to occur directly with memory
  - The processor grants to an I/O module the authority to read from or write to memory so that the I/O memory transfer can occur without tying up the processor
  - The I/O module issues read or write commands to memory relieving the processor of responsibility for the exchange
  - This operation is known as direct memory access (DMA)

Interconnection Structures



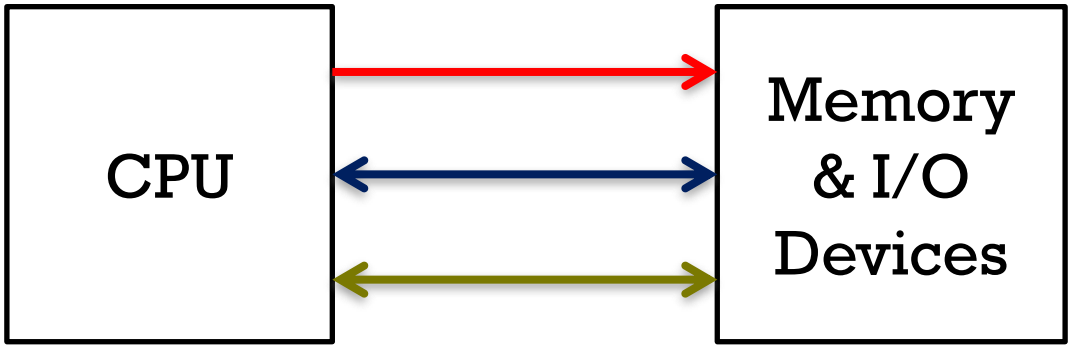




+ Other Components

- System Buses
  - Collection of communication line components of computers

  - Types:
    - Address Bus
    - Data Bus
    - Control Bus



  - Memory Read/Write

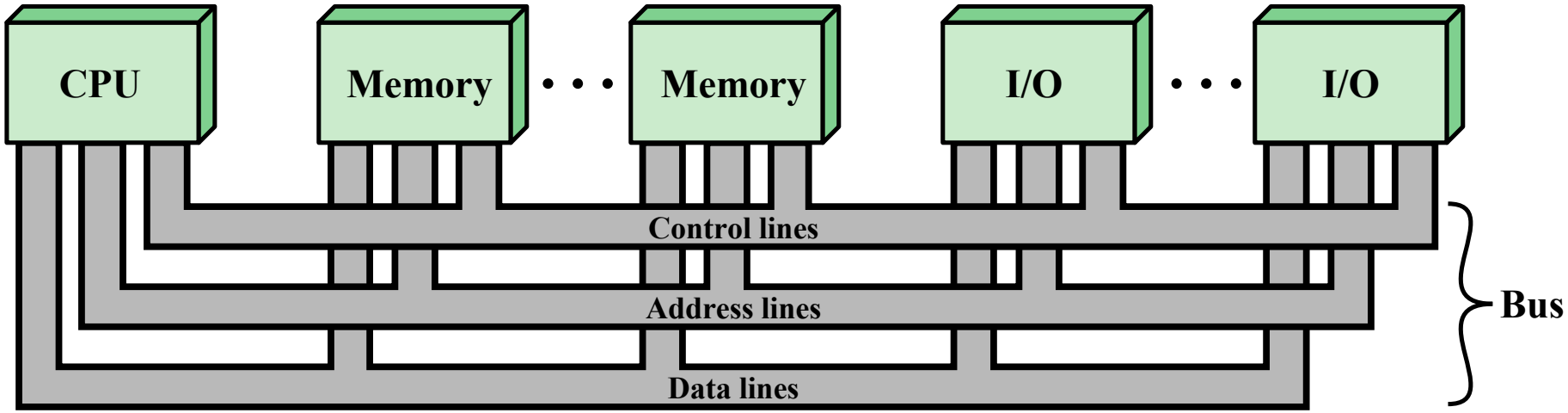


Figure 3.16 Bus Interconnection Scheme

+ Other Components

- CPU Registers
  - Accumulator
  - Program Counter
  - .
  - .
  - .
- Types of Architecture
  - AC-based Architecture
  - Register-based Architecture
  - Register-Memory-based Architecture
  - Complex-Memory-based Architecture
  - Stack-based Architecture

Micro-operations

- The operation executed on the value stored on registers
- Symbolic notation to describe the micro-operations:
  Register Transfer language (RTL).
- Transfer Notations:
  R1→ R2 or R2← R1

- Parallel Operation Notation: R1← R2, PC ← PC+1
  "These two operations can be executed simultaneously if a set of components required by these two micro-operations are mutually exclusive."
- Memory Transfer:
  - Read: Memory to CPU
    R1 ← M[address], R1 ← M[100] or R1 ← M[AR]
  - Write: CPU to Memory
    M1[100] ← R1 or M[AR] ← R1

Questions

- If the content of register R1 is 5 and the content of memory at address 1000 is 20, then the content of Register R2 after the following code execution?

  R1 ← R1+1

  R2 ← R1+M[1000]

Instructions

- A group of bits which instructs the computer to perform kind of operation.

  High level Instructions - Low level Instructions

  (C, Python, Java, etc.) (Byte code, Machine code, etc.)

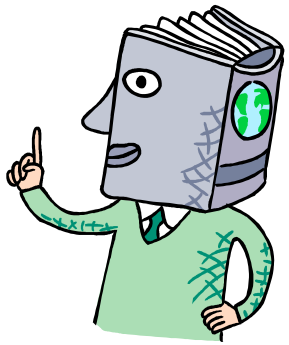
Opcode | Operands

Opcode | Address 1 | Address 2 | Address 3

- Instruction Formats:
  - 3-Address Instruction
  - 2-Address Instruction
  - 1-Address Instruction
  - 0-Address Instruction

Machine Instruction Characteristics

- The operation of the processor is determined by the instructions it executes, referred to as *machine instructions* or *computer instructions*
- The collection of different instructions that the processor can execute is referred to as the processor's *instruction set*
- Each instruction must contain the information required by the processor for execution



Elements of a Machine Instruction

Operation code (opcode)

- Specifies the operation to be performed. The operation is specified by a binary code, known as the operation code, or *opcode*

Source operand reference

- The operation may involve one or more source operands, that is, operands that are inputs for the operation

Result operand reference

- The operation may produce a result

Next instruction reference

- This tells the processor where to fetch the next instruction after the execution of this instruction is complete

Source and result operands can be in one of four areas:

1) Main or virtual memory
   - As with next instruction references, the main or virtual memory address must be supplied

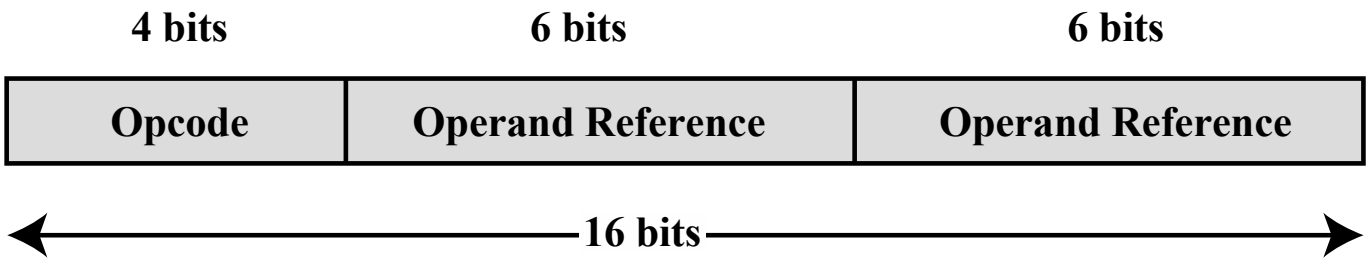
2) I/O device
   - The instruction must specify the I/O module and device for the operation. If memory-mapped I/O is used, this is just another main or virtual memory address

3) Processor register
   - A processor contains one or more registers that may be referenced by machine instructions.
   - If more than one register exists each register is assigned a unique name or number and the instruction must contain the number of the desired register

4) Immediate
   - The value of the operand is contained in a field in the instruction being executed

Instruction Representation

- Within the computer each instruction is represented by a sequence of bits
- The instruction is divided into fields, corresponding to the constituent elements of the instruction



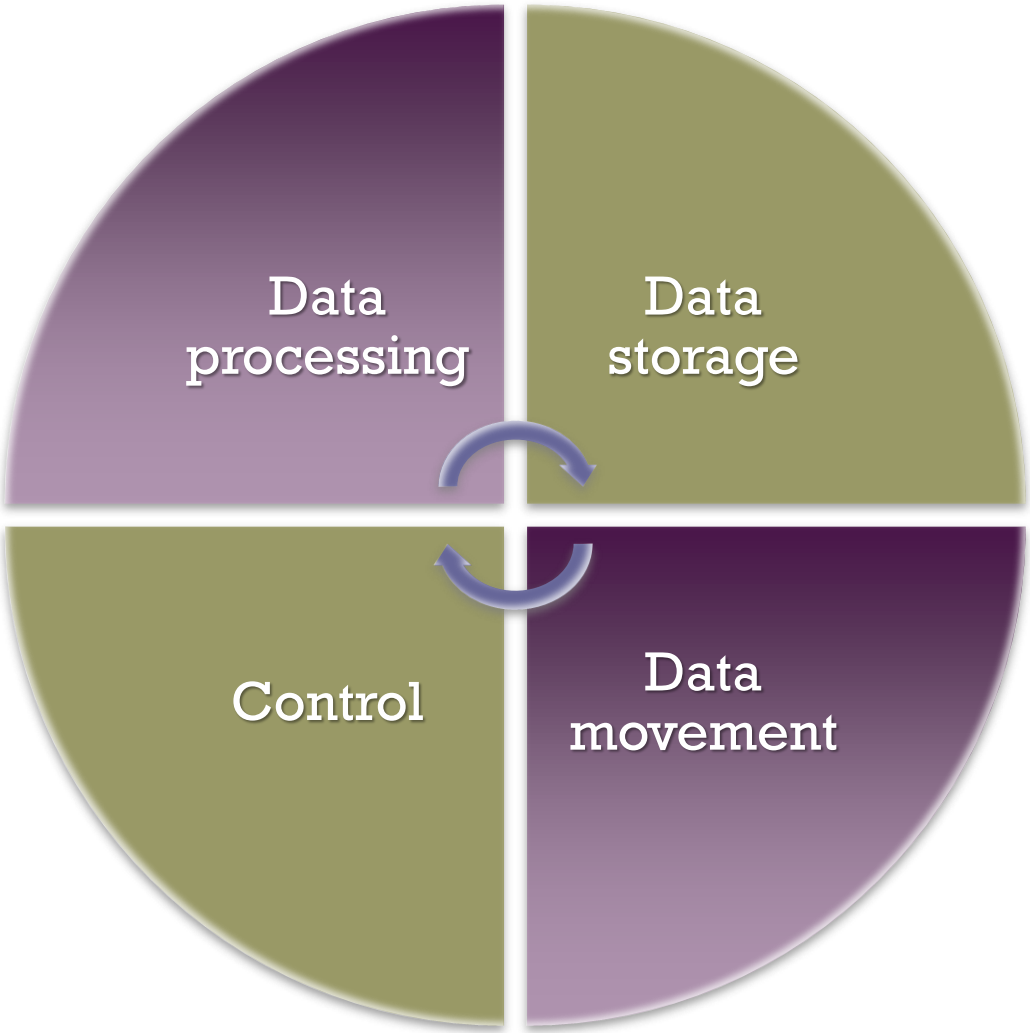
How much memory can be referenced by this instruction?

Figure 12.2 A Simple Instruction Format

Instruction Representation

- Opcodes are represented by abbreviations called *mnemonics*
- Examples include:
  - ADD Add
  - SUB Subtract
  - MUL Multiply
  - DIV Divide
  - LOAD Load data from memory
  - STOR Store data to memory
- Operands are also represented symbolically
- Each symbolic opcode has a fixed binary representation
  - The programmer specifies the location of each symbolic operand

Instruction Types



Instruction		Comment
SUB	Y, A, B	Y ¬ A – B
MPY	T, D, E	T ¬ D ´ E
ADD	T, T, C	T ¬ T + C
DIV	Y, Y, T	Y ¬ Y ÷ T

(a) Three-address instructions

Instruction		Comment
MOVE	Y, A	Y ¬ A
SUB	Y, B	Y ¬ Y – B
MOVE	T, D	T ¬ D
MPY	T, E	T ¬ T ´ E
ADD	T, C	T ¬ T + C
DIV	Y, T	Y ¬ Y ÷ T

(b) Two-address instructions

Instruction		Comment
LOAD	D	AC ¬ D
MPY	E	AC ¬ AC ´ E
ADD	C	AC ¬ AC + C
STOR	Y	Y ¬ AC
LOAD	A	AC ¬ A
SUB	B	AC ¬ AC – B
DIV	Y	AC ¬ AC ÷ Y
STOR	Y	Y ¬ AC

(c) One-address instructions

Figure 12.3 Programs to Execute $Y = \frac{A - B}{C + (D \times E)}$

Utilization of Instruction Addresses (Non-branching Instructions)

Number of Addresses	Symbolic Representation	Interpretation
3	OP A, B, C	A ¬ B OP C
2	OP A, B	A ¬ A OP B
1	OP A	AC ¬ AC OP A
0	OP	T ¬ (T – 1) OP T

AC = accumulator
T = top of stack
(T – 1) = second element of stack
A, B, C = memory or register locations

Questions

- Consider a digital computer that supports only 2-address instructions, each with 16 bits. If the address is 5 bits, then maximum and minimum, how many instructions can the systems support?

- Consider a digital computer that supports 16 2-address instructions. If the address length is 6-bits, then the length of the instruction is _____ bits.

- A processor has 40 distinct instructions and 24 general-purpose registers. A 32-bit instruction word has an opcode, two register operands, and an immediate operand. The number of bits available for the immediate operand field is _____.

Questions

- Simplified Computer Instructions
  - OP Rj, Ri
  - OP m, Ri
  - MOV m, Ri
  - MOV Ri, m

- Computer has two registers and performs two operations add and SUB. Identify the minimum number of MOV operations required in the following basic block. All the operands are stored in memory, and output must be stored in memory.

  t1 = a+b
  t2 = c+d
  t3 = e-t2
  t4 = t1-t2

Instruction Set Design

Very complex because it affects so many aspects of the computer system

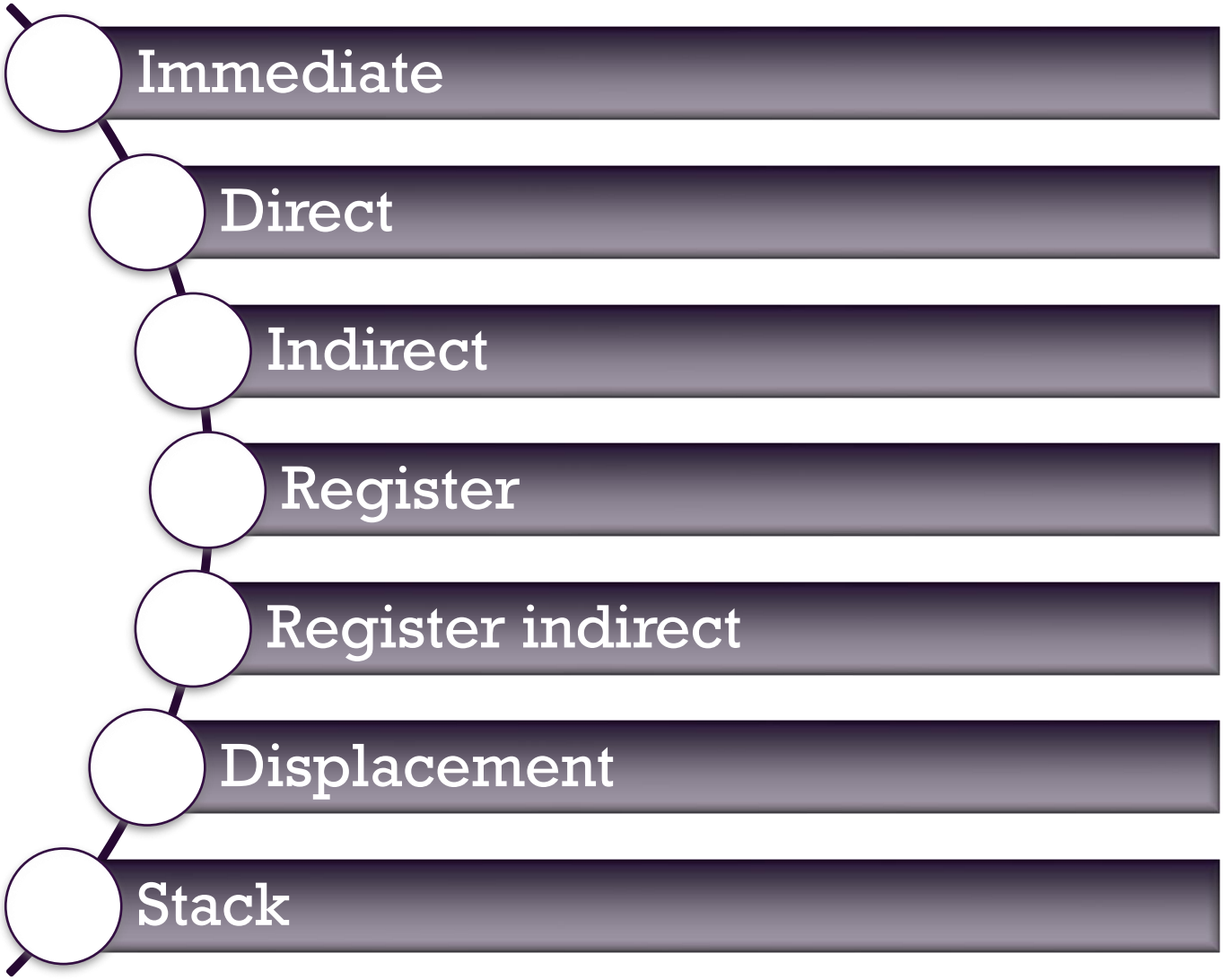
Defines many of the functions performed by the processor

Programmer's means of controlling the processor

Fundamental design issues:

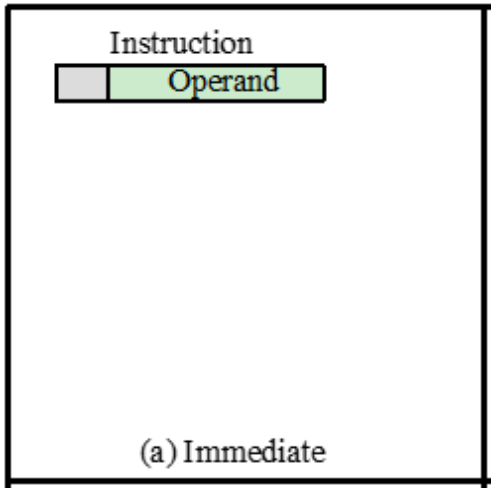
Operation repertoire	Data types	Instruction format	Registers	Addressing
• How many and which operations to provide and how complex operations should be	• The various types of data upon which operations are performed	• Instruction length in bits, number of addresses, size of various fields, etc.	• Number of processor registers that can be referenced by instructions and their use	• The mode or modes by which the address of an operand is specified

Addressing Modes



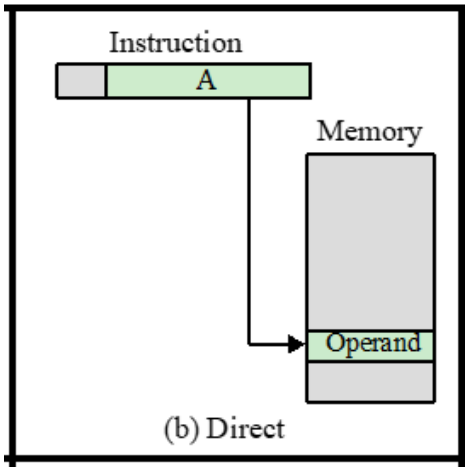
Immediate Addressing

- Simplest form of addressing
- Operand = A
- This mode can be used to define and use constants or set initial values of variables
  - Typically, the number will be stored in two complement form
  - The leftmost bit of the operand field is used as a sign bit
- Advantage:
  - No memory reference other than the instruction fetch is required to obtain the operand, thus saving one memory or cache cycle in the instruction cycle
- Disadvantage:
  - The size of the number is restricted to the size of the address field, which, in most instruction sets, is small compared with the word length



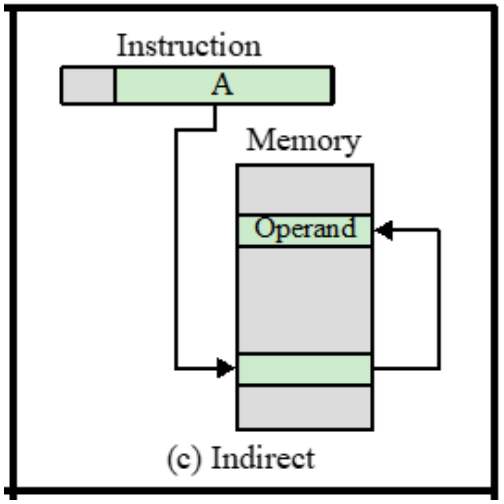
Direct Addressing

- Address field contains the effective address of the operand
- Effective address (EA) = address field (A)
- Requires only one memory reference and no special calculation
- Limitation
  - it provides only a limited address space



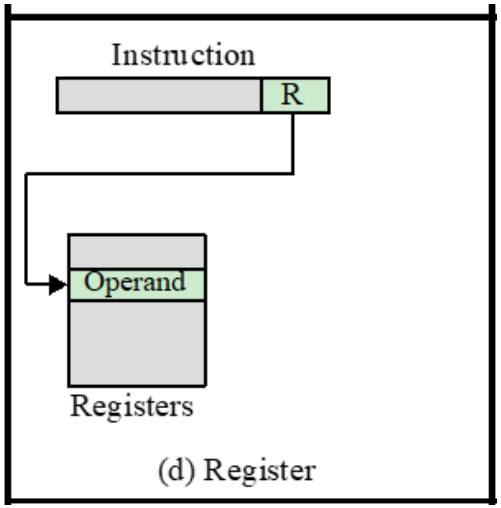
Indirect Addressing

- Reference to the address of a word in memory that contains a full-length address of the operand
- EA = (A)
  - Parentheses are to be interpreted as meaning *contents of*
- Advantage:
  - For a word length of *N* an address space of 2^N is now available
- Disadvantage:
  - Instruction execution requires two memory references to fetch the operand
    - One to get its address and a second to get its value
- A rarely used variant of indirect addressing is multilevel or cascaded indirect addressing
  - EA = (. . . (A) . . .)
  - Disadvantage is that three or more memory references could be required to fetch an operand



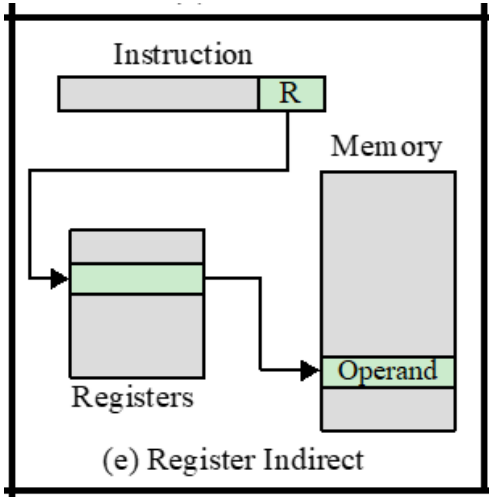
Register Addressing

- Address field refers to a register rather than a main memory address
- EA = R
- Advantages:
  - Only a small address field is needed in the instruction
  - No time-consuming memory references are required
- Disadvantage:
  - The address space is very limited



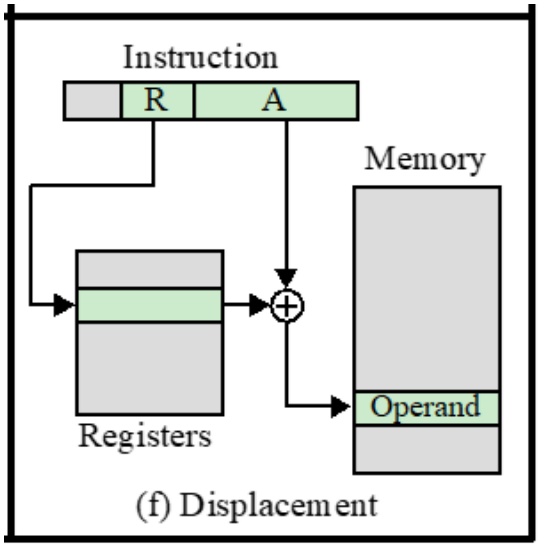
Register Indirect Addressing

- Analogous to indirect addressing
  - The only difference is whether the address field refers to a memory location or a register
- EA = (R)
- Address space limitation of the address field is overcome by having that field refer to a word-length location containing an address
- Uses one less memory reference than indirect addressing



Displacement Addressing

- Combines the capabilities of direct addressing and register indirect addressing
- EA = A + (R)
- Requires that the instruction have two address fields, at least one of which is explicit
  - The value contained in one address field (value = A) is used directly
  - The other address field refers to a register whose contents are added to A to produce the effective address
- Most common uses:
  - Relative addressing
  - Base-register addressing
  - Indexing



+Relative Addressing

The implicitly referenced register is the program counter (PC)

- The next instruction address is added to the address field to produce the EA
- Typically the address field is treated as a twos complement number for this operation
- Thus the effective address is a displacement relative to the address of the instruction

Exploits the concept of locality

Saves address bits in the instruction if most memory references are relatively near to the instruction being executed

+Base-Register Addressing

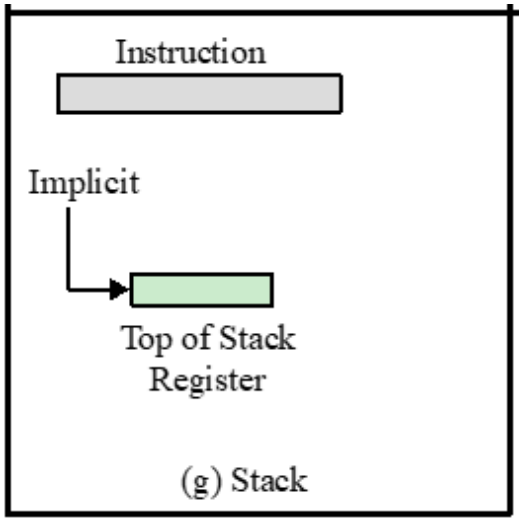
- The referenced register contains a main memory address, and the address field contains a displacement from that address
- The register reference may be explicit or implicit
- Exploits the locality of memory references
- Convenient means of implementing segmentation
- In some implementations, a single segment base register is employed and is used implicitly
- In others the programmer may choose a register to hold the base address of a segment and the instruction must reference it explicitly

+Indexing

- The address field references a main memory address, and the referenced register contains a positive displacement from that address
- The method of calculating the EA is the same as for base-register addressing
- An important use is to provide an efficient mechanism for performing iterative operations
- Autoindexing
  - Automatically increment or decrement the index register after each reference to it
  - EA = A + (R)
  - (R) ← (R) + 1
- Postindexing
  - Indexing is performed after the indirection
  - EA = (A) + (R)
- Preindexing
  - Indexing is performed before the indirection
  - EA = (A + (R))

Stack Addressing

- A stack is a linear array of locations
  - Sometimes referred to as a *pushdown list* or *last-in-first-out queue*
- A stack is a reserved block of locations
  - Items are appended to the top of the stack so that the block is partially filled
- Associated with the stack is a pointer whose value is the address of the top of the stack
  - The stack pointer is maintained in a register
  - Thus references to stack locations in memory are in fact register indirect addresses
- Is a form of implied addressing
- The machine instructions need not include a memory reference but implicitly operate on the top of the stack



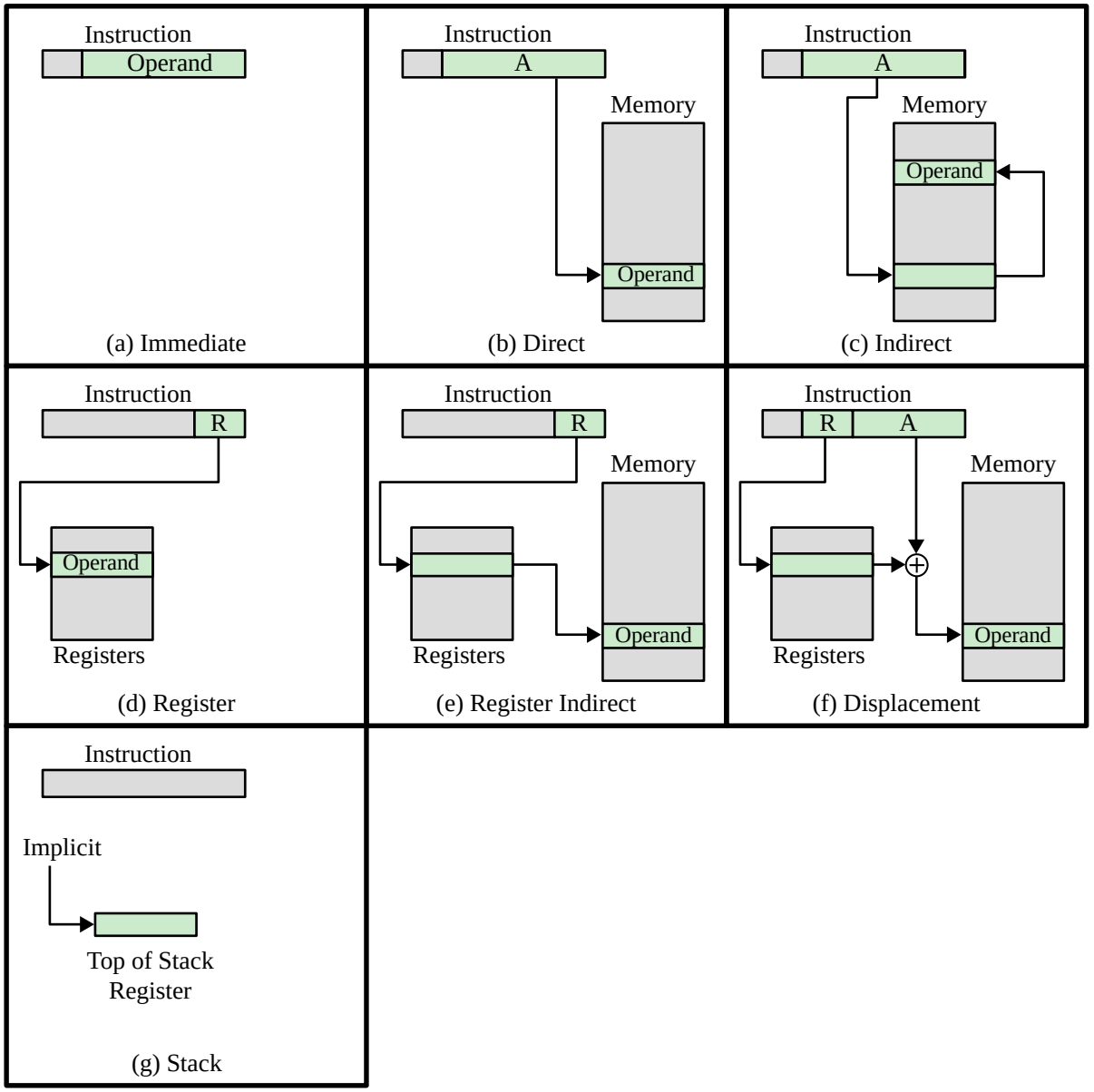


Figure 13.1 Addressing Modes

Table 13.1
Basic Addressing Modes

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

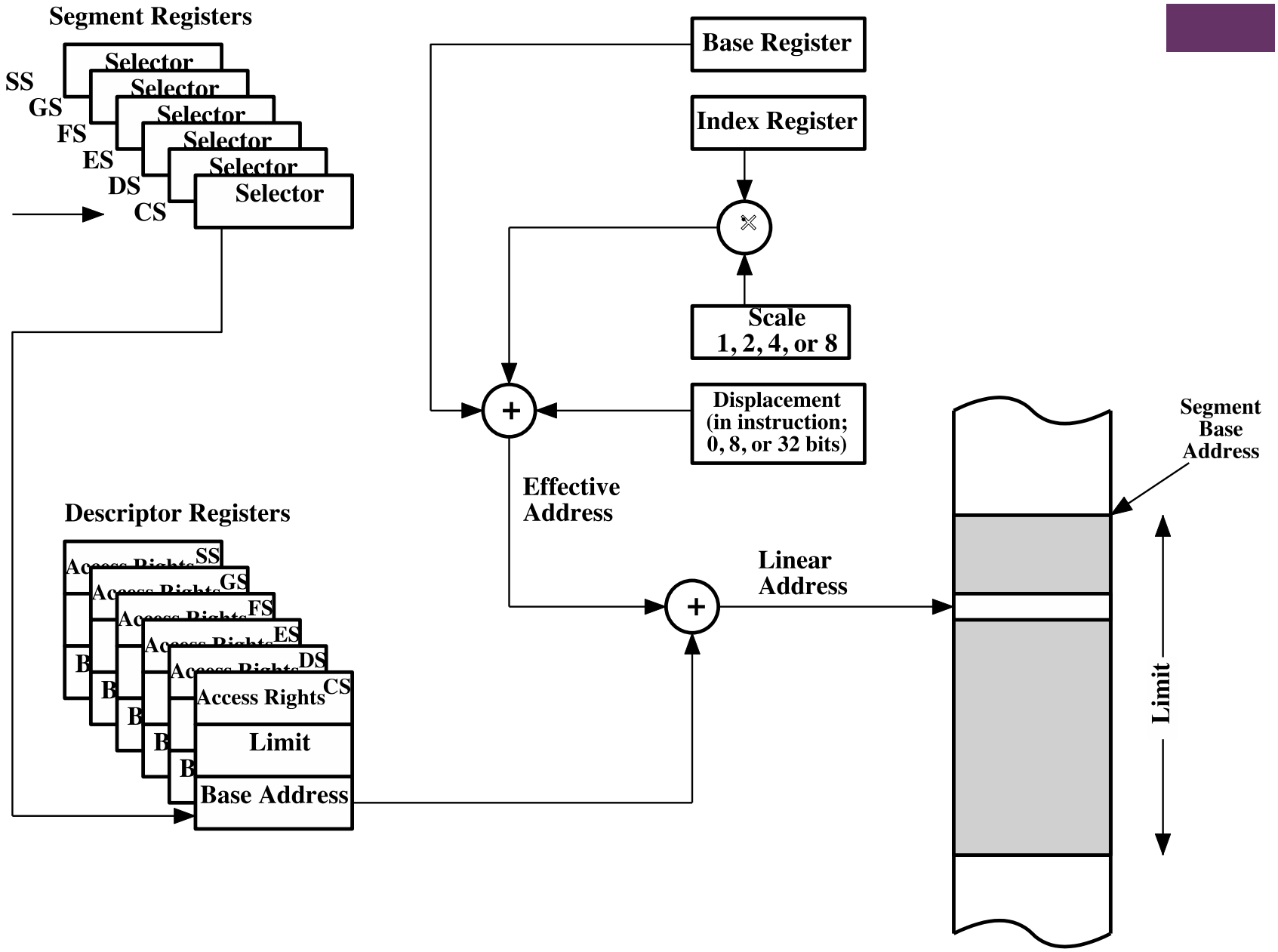


Figure 13.2 x86 Addressing Mode Calculation

Table 13.2
x86 Addressing Modes

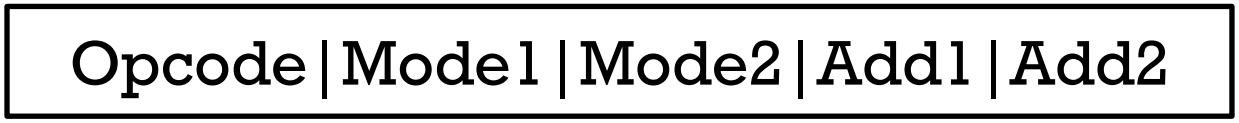
Mode	Algorithm
Immediate	Operand = A
Register Operand	LA = R
Displacement	LA = (SR) + A
Base	LA = (SR) + (B)
Base with Displacement	LA = (SR) + (B) + A
Scaled Index with Displacement	LA = (SR) + (I) ´ S + A
Base with Index and Displacement	LA = (SR) + (B) + (I) + A
Base with Scaled Index and Displacement	LA = (SR) + (I) ´ S + (B) + A
Relative	LA = (PC) + A

LA = linear address
(X) = contents of X
SR = segment register
PC = program counter
A = contents of an address field in the instruction
R = register
B = base register
I = index register
S = scaling factor

Questions

- What happens if the instruction has two address

- Solution



- If the first operand comes from the register then only one addressing mode is required.

Questions

	Memory
200	Opcode | Mode
201	Address =500
202	Next Instruction
399	450
400	700
500	800
600	900
702	Target Instruction
800	300

PC = 202 | R1 = 400 | XR = 100 | AC

Mode	E.A.	Operand
Immediate	-	500
Direct	500	800
Indirect	800	300
Register	-	400
Register Indirect	400	700
Autodecrement	399	450
Indexed	600	900
PC-Relative	702	-

Instruction Formats

Define the layout of the bits of an instruction, in terms of its constituent fields

Must include an opcode and, implicitly or explicitly, indicate the addressing mode for each operand

For most instruction sets more than one instruction format is used

Instruction Length

- Most basic design issue
- Affects, and is affected by:
  - Memory size
  - Memory organization
  - Bus structure
  - Processor complexity
  - Processor speed
- Should be equal to the memory-transfer length or one should be a multiple of the other
- Should be a multiple of the character length, which is usually 8 bits, and of the length of fixed-point numbers

Allocation of Bits

- Number of addressing modes
- Number of operands
- Register versus memory
- Number of register sets
- Address range
- Address granularity

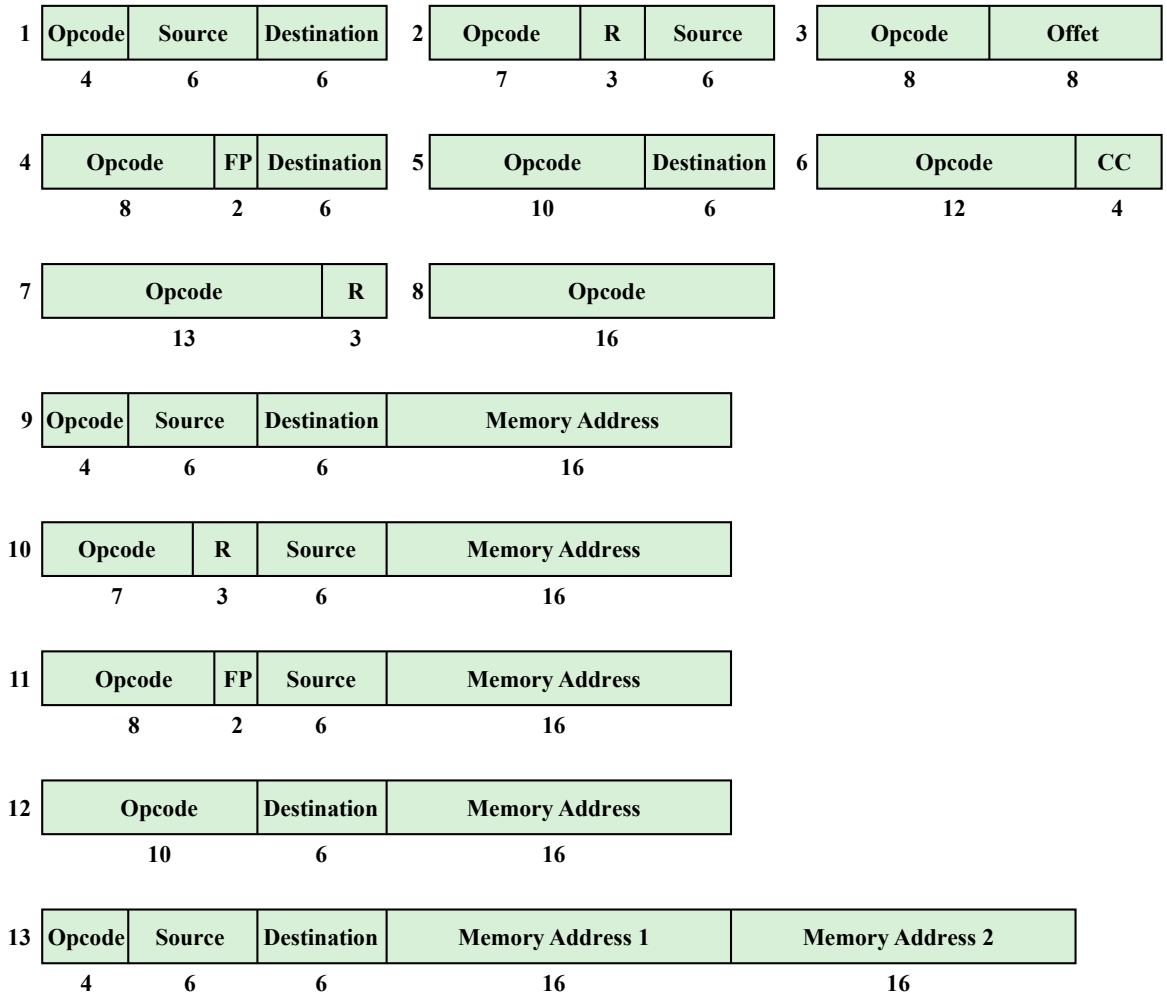
Instructions Length

- Fixed length Instructions
  - Variable length opcode
- Variable length Instructions
  - Fixed length opcode

Questions

- Consider a system that supports 2-address instructions and 1-address instructions. The system has 6-bit instructions and a 2-bit address. If there are three 2-address instructions in the design, then maximum and minimum how many 1-address instructions the systems can support?

- Consider a system with 24-bit instructions and 9-bit addresses. If there are 60 2-address instructions, then the maximum how many 1-address instructions can be formulated in the system?



Numbers below fields indicate bit length
Source and Destination each contain a 3-bit addressing mode field and a 3-bit register number
FP indicates one of four floating-point registers
R indicates one of the general-purpose registers
CC is the condition code field

Figure 13.7 Instruction Formats for the PDP-11

Address	Contents			
101	0010	0010	101	2201
102	0001	0010	102	1202
103	0001	0010	103	1203
104	0011	0010	104	3204
201	0000	0000	201	0002
202	0000	0000	202	0003
203	0000	0000	203	0004
204	0000	0000	204	0000

(a) Binary program

Address	Contents
101	2201
102	1202
103	1203
104	3204
201	0002
202	0003
203	0004
204	0000

(b) Hexadecimal program

Address	Instruction	
101	LDA	201
102	ADD	202
103	ADD	203
104	STA	204
201	DAT	2
202	DAT	3
203	DAT	4
204	DAT	0

(c) Symbolic program

Label	Operation	Operand
FORMUL	LDA	I
	ADD	J
	ADD	K
	STA	N
I	DATA	2
J	DATA	3
K	DATA	4
N	DATA	0

(d) Assembly program

Figure 13.14 Computation of the Formula N = I + J+ K