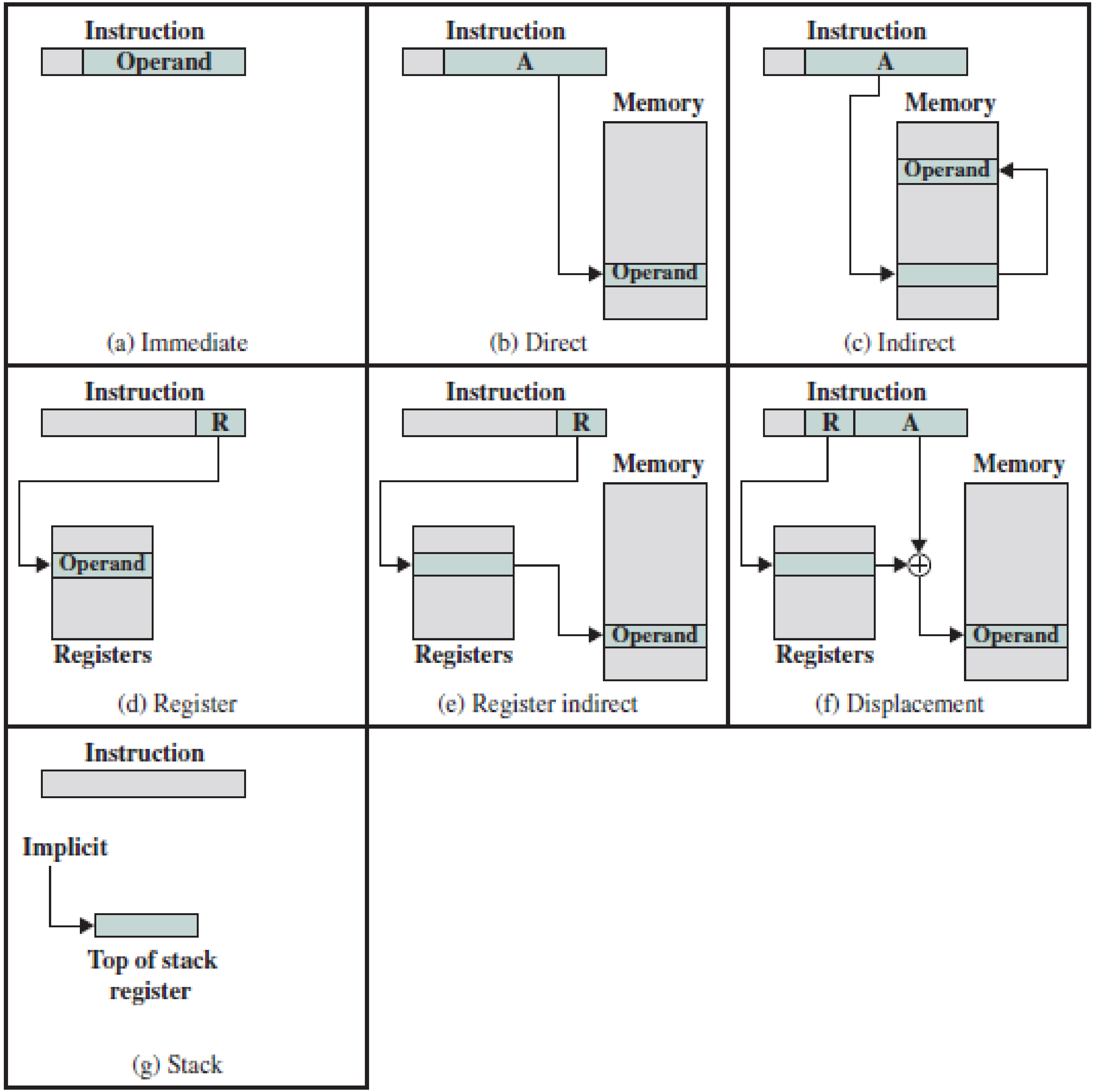
ADDRESSING MODES AND FORMATS

ADDRESSING MODES

- The most common addressing techniques, or modes are:
  - Immediate
  - Direct
  - Indirect
  - Register
  - Register indirect
  - Displacement
  - Stack
- Notation used are:
  - A = contents of an address field in the instruction
  - R = contents of an address field in the instruction that refers to a register
  - EA = actual (effective) address of the location containing the referenced operand
  - (X) = contents of memory location X or register X

Addressing Modes



Addressing Modes

- All computer architectures provide more than one of these addressing modes. The question arises as to how the processor can determine which address mode is being used in a particular instruction.
  - Different opcodes will use different addressing modes.
  - One or more bits in the instruction format can be used as a mode field.
- 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.

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

ASSEMBLY LANGUAGE

- A processor can understand and execute machine instructions. If a programmer wished to program directly in machine language, then it would be necessary to enter the program as binary data.
- Consider the simple BASIC statement

$$N = I + J + K$$

- Suppose we wished to program this statement in machine language and to initialize I, J, and K to 2, 3, and 4, respectively. The program starts in location 101 (hexadecimal). Memory is reserved for the four variables starting at location 201.

ASSEMBLY LANGUAGE

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

ASSEMBLY LANGUAGE

- Programs written in assembly language (assembly programs) are translated into machine language by an assembler.
- This program must not only do the symbolic translation discussed earlier but also assign some form of memory addresses to symbolic addresses.

In computer programming, assembly language is any low-level programming language with a very strong correspondence between the instructions in the language and the architecture's machine code instructions.