

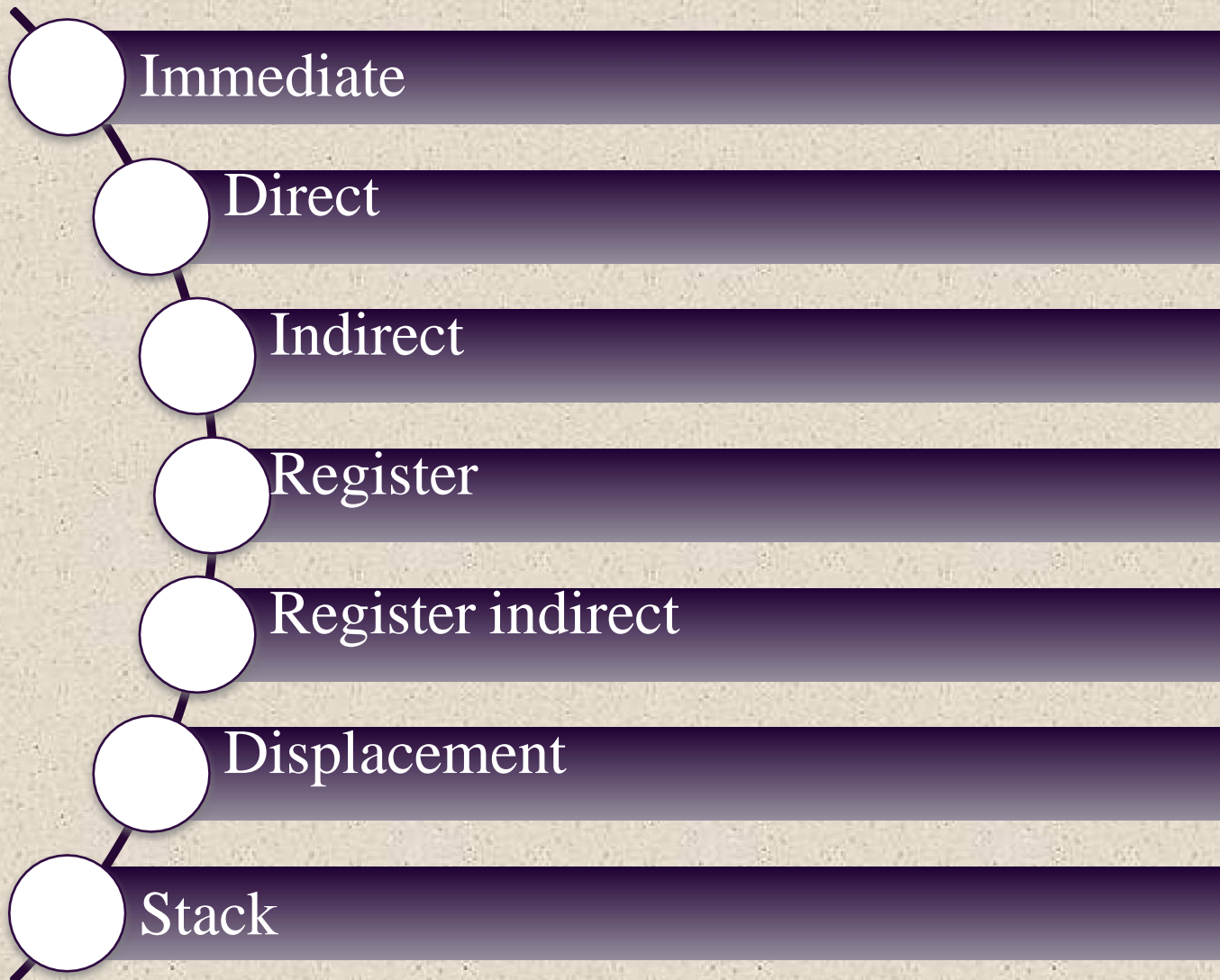
William Stallings
Computer Organization
and Architecture
10th Edition



+ Chapter 13

Instruction Sets: Addressing Modes and Formats

Addressing Modes



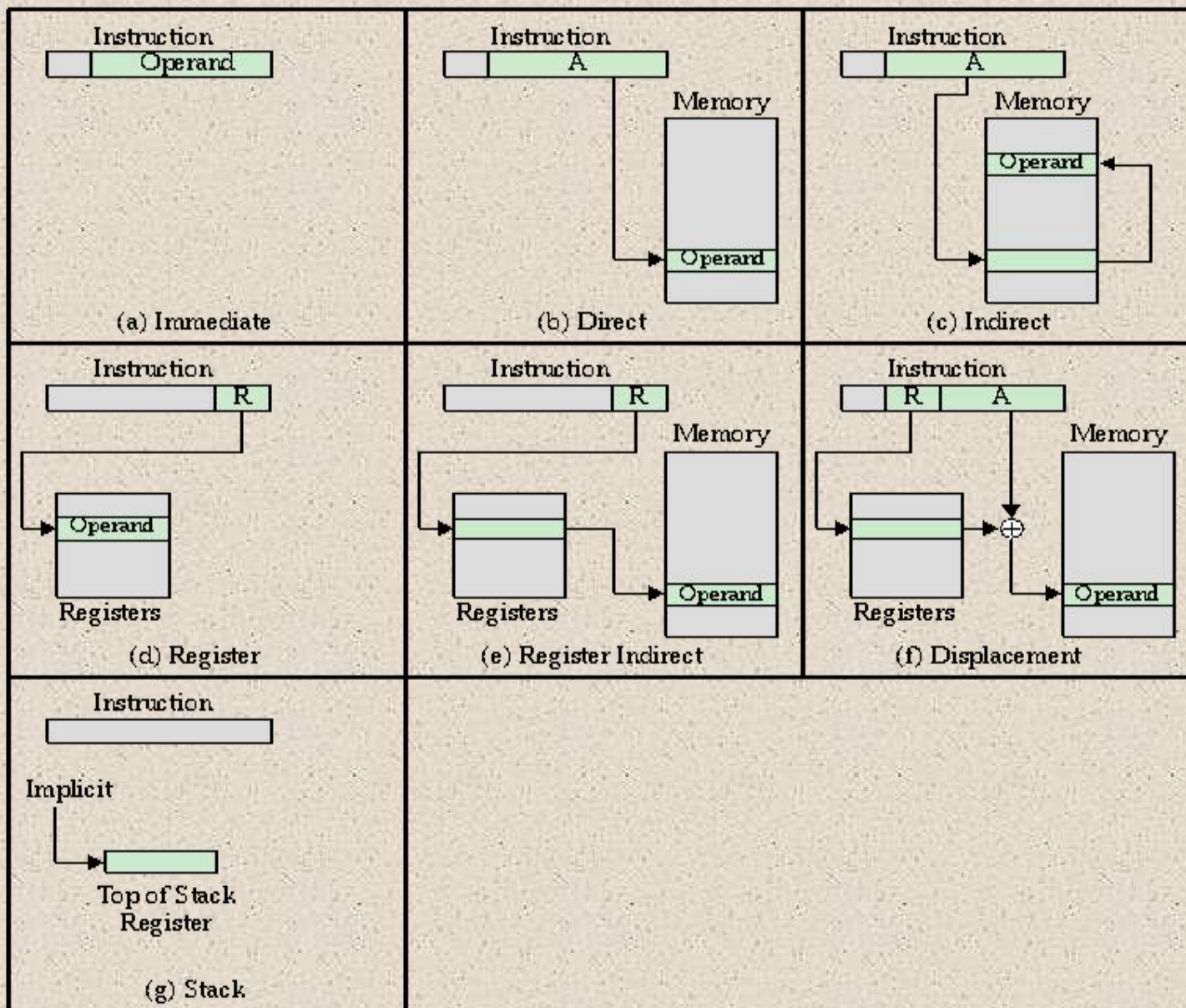


Figure 13.1 Addressing Modes




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

+ 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 twos 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)

Was common in earlier generations of computers

Requires only one memory reference and no special calculation

Limitation is that it provides only a limited address space



+ Indirect Addressing



- Reference to the address of a word in memory which 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
 - $EA = (\dots (A) \dots)$
 - 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 two's 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
 - $EA = A + (R)$
 - $(R) \leftarrow (R) + 1$
- 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 implicitly operate on the top of the stack



Table 13.2

x86 Addressing Modes

Mode	Algorithm
Immediate	$\text{Operand} = A$
Register Operand	$\text{LA} = R$
Displacement	$\text{LA} = (\text{SR}) + A$
Base	$\text{LA} = (\text{SR}) + (B)$
Base with Displacement	$\text{LA} = (\text{SR}) + (B) + A$
Scaled Index with Displacement	$\text{LA} = (\text{SR}) + (I) \times S + A$
Base with Index and Displacement	$\text{LA} = (\text{SR}) + (B) + (I) + A$
Base with Scaled Index and Displacement	$\text{LA} = (\text{SR}) + (I) \times S + (B) + A$
Relative	$\text{LA} = (\text{PC}) + A$

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