William Stallings
Computer Organization
and Architecture
7th Edition

Chapter 11

Instruction Sets:

Addressing Modes and Formats

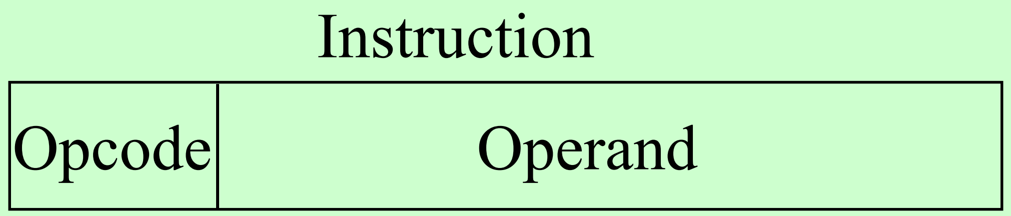
Addressing Modes

- Immediate
- Direct
- Indirect
- Register
- Register Indirect
- Displacement (Indexed)
- Stack

Immediate Addressing

- Operand is part of instruction
- Operand = address field
- e.g. ADD 5
  - —Add 5 to contents of accumulator （*only need 1 address*）
  - —5 is operand
- + No memory reference to fetch data
- + Fast
- - Limited range

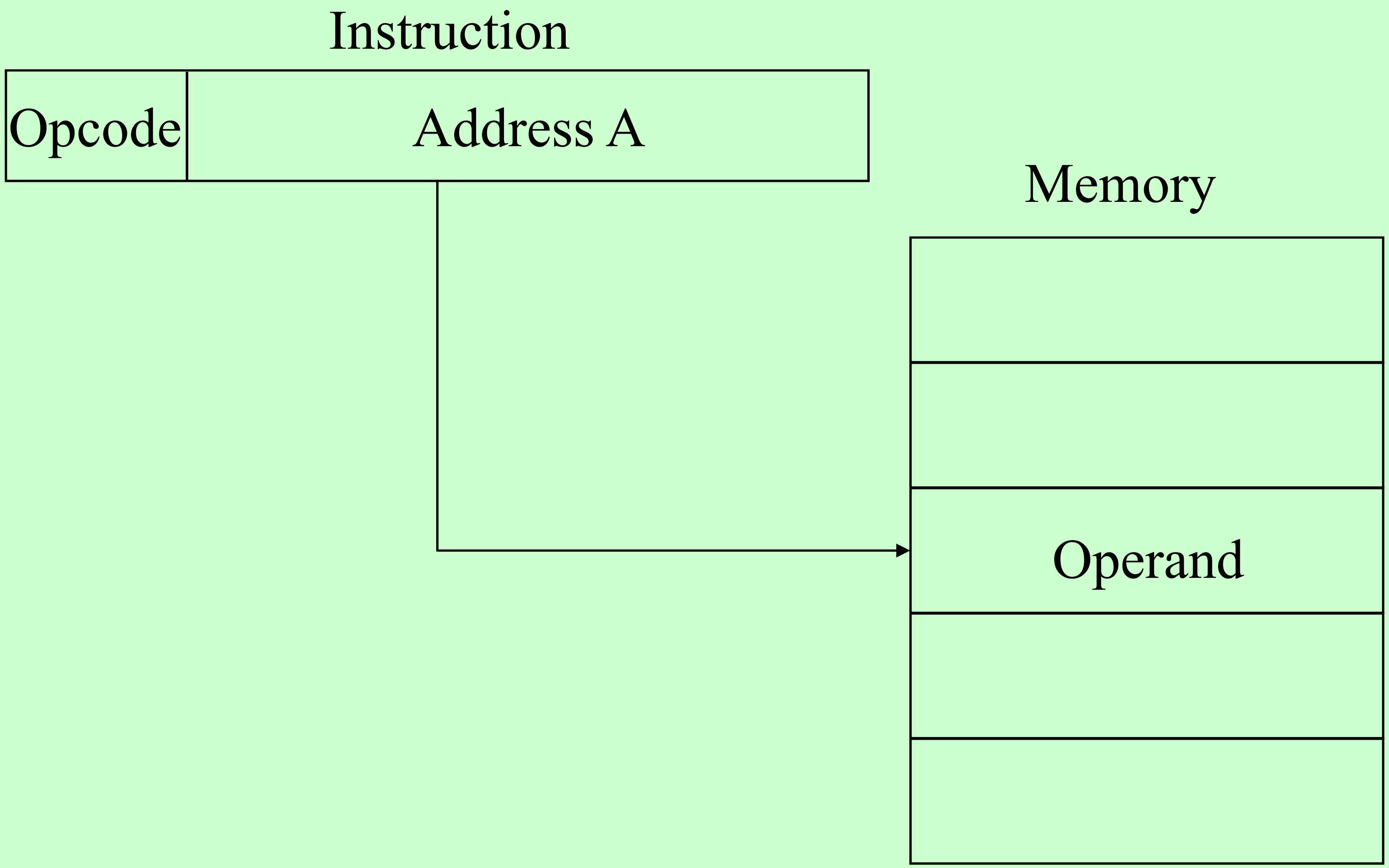
Immediate Addressing Diagram



Direct Addressing

- Address field contains address of operand
- *Effective address* EA = *address field* A
- e.g. ADD A
  - Add contents of *cell A* to accumulator
  - Look in memory at address A for operand
- +Single memory reference to access data
- +No additional calculations to *work out* effective address
- - Limited address space

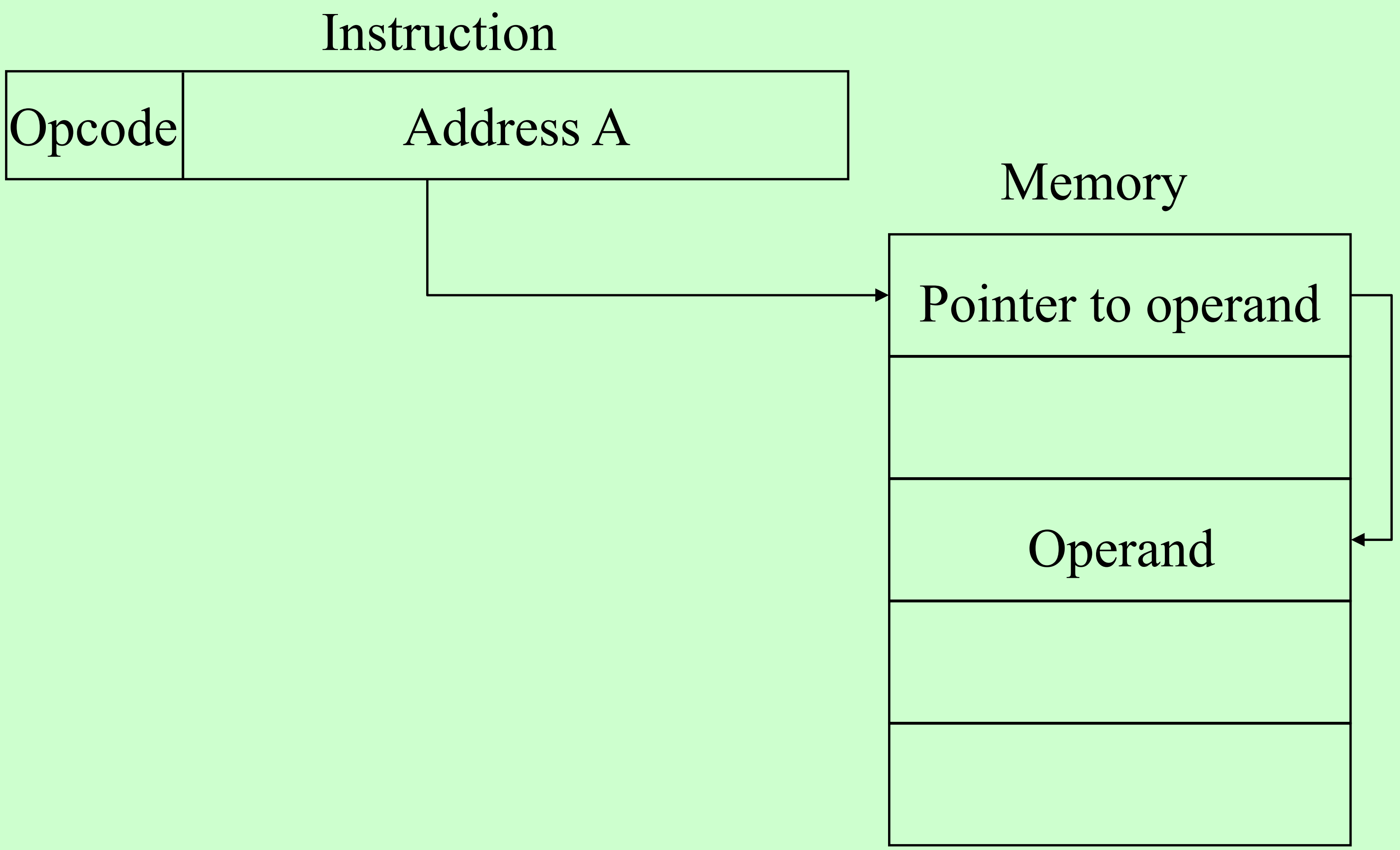
Direct Addressing Diagram



Indirect Addressing (1)

- Memory cell pointed to by address field contains the *address (pointer)* of the operand
- EA = (A)
  - Look in A, find address A and look there for operand
- e.g. ADD (A)
  - Add contents of cell pointed to by contents of A to accumulator

Indirect Addressing Diagram



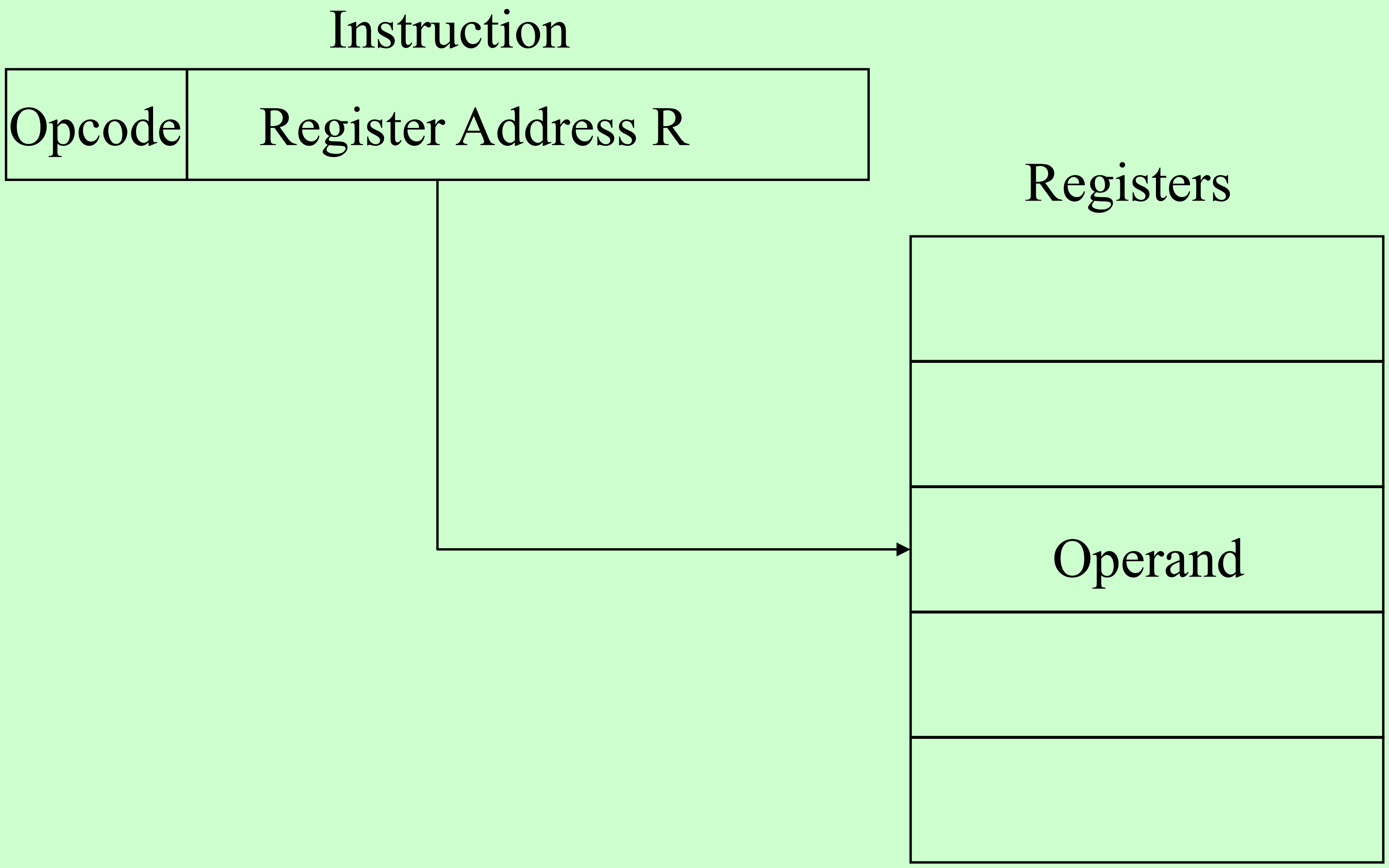
Indirect Addressing (2)

- + Large address space
- + 2^n where n = word length
- May be nested, multilevel, cascaded
  - e.g. EA = (((A)))
    - Draw the diagram yourself
- - Multiple memory accesses to find operand （*more than 1 time*）
- - Hence slower

Register Addressing (1)

- Operand is held in register named in address filed
- EA = R （*3~5bits*）
- Limited number of registers （*8~32*）
- Very small address field needed
  - —Shorter instructions
  - —Faster instruction fetch
  - —No need to access memory

Register Addressing Diagram



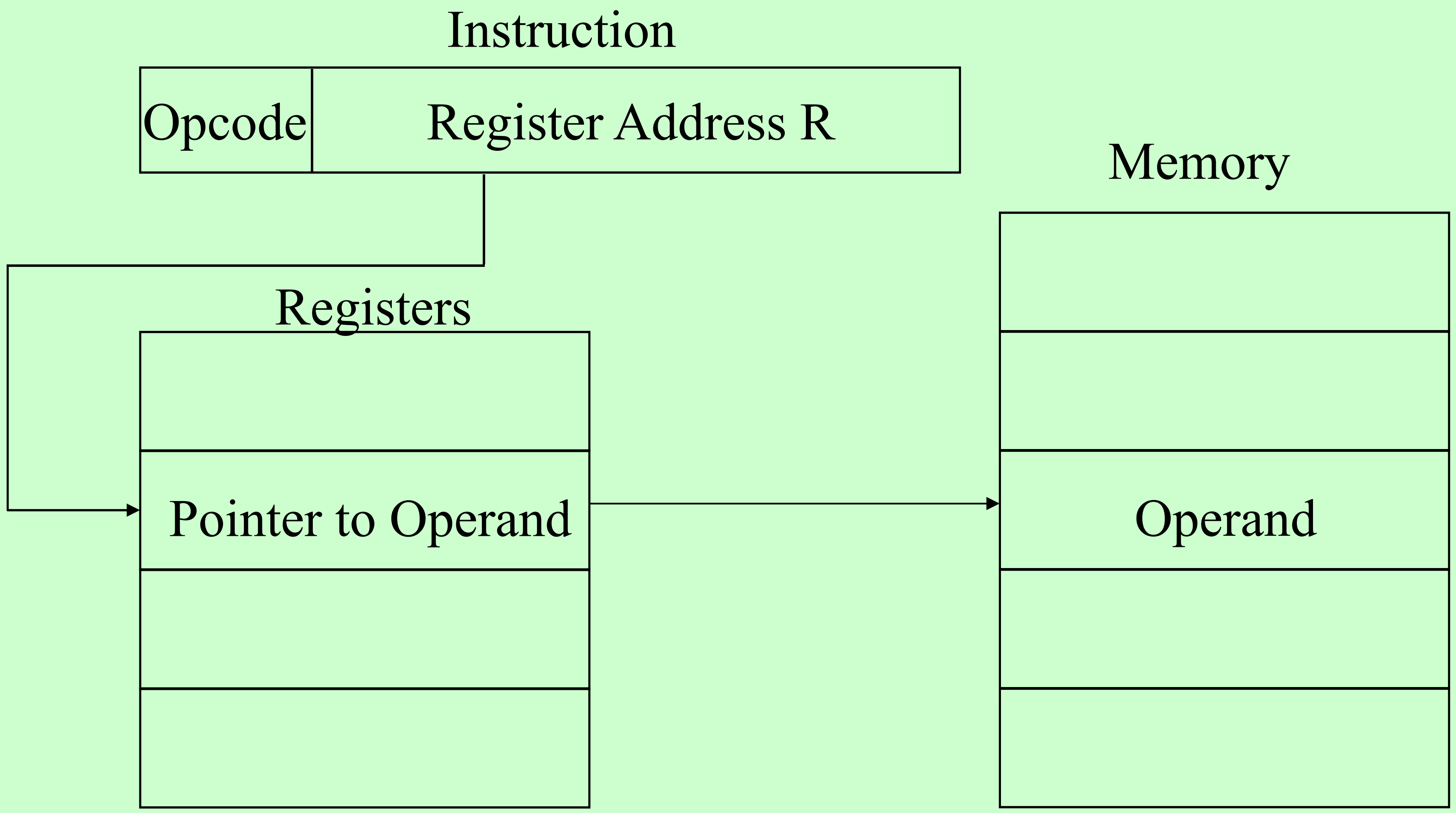
Register Addressing (2)

- +No memory access
- +Very fast execution
- - Very limited address space
- Multiple registers helps performance
  - Requires good assembly programming or compiler writing
  - C programming
    - register int a;
- c.f. Direct addressing

Register Indirect Addressing

- C.f. indirect addressing
- EA = (R)
- Operand is in *memory cell* pointed to by contents of register R
- Large address space (2^n)
- *One fewer memory access* than indirect addressing

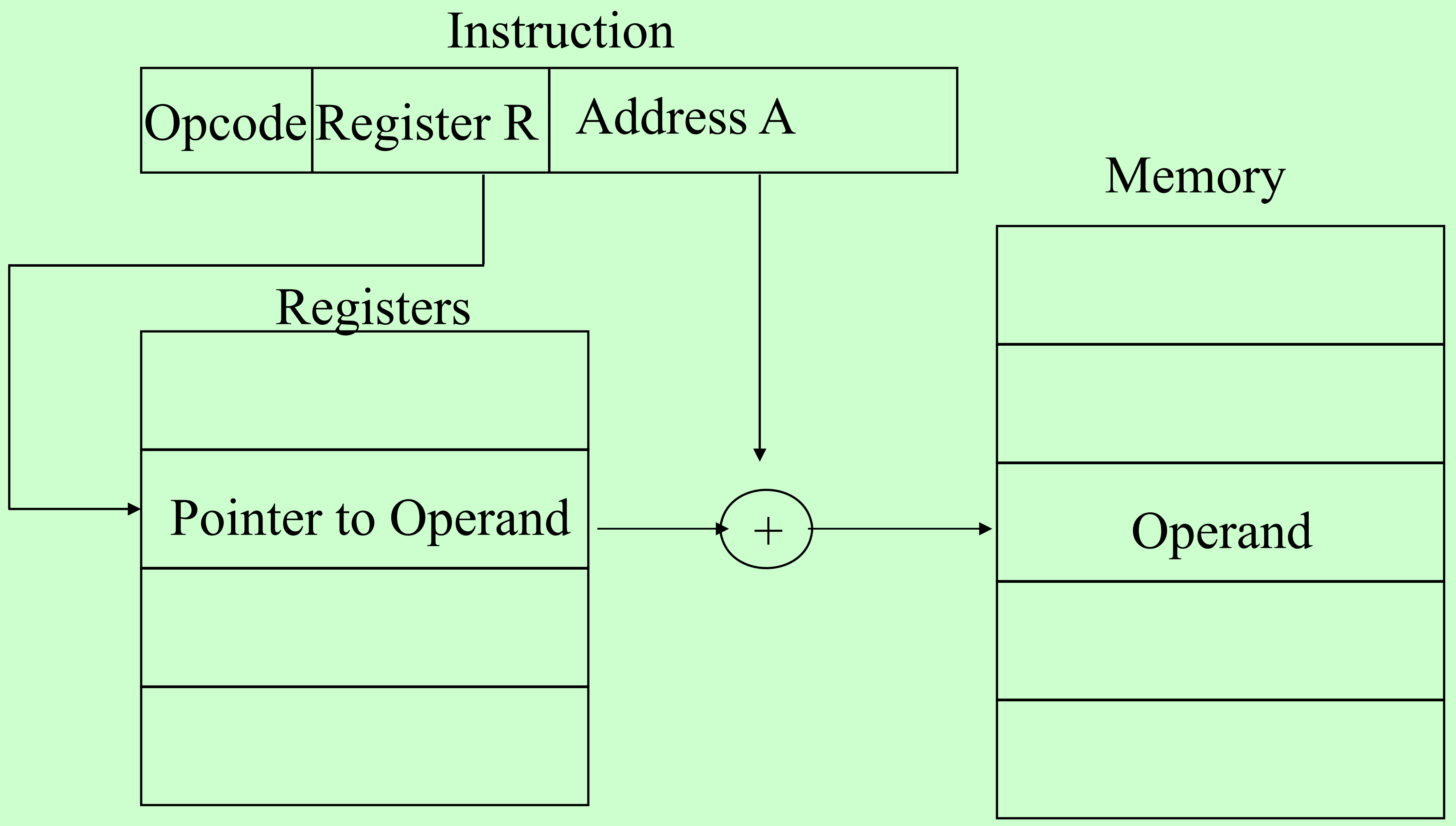
Register Indirect Addressing Diagram



Displacement Addressing

- EA = A + (R)
- Address field hold two values
  - A = base value
  - R = register that holds displacement
  - or vice versa

Displacement Addressing Diagram



Relative Addressing

- A version of displacement addressing
- R = *Program counter*, PC
- EA = A + (PC)
- i.e. get operand *from A cells* from current location pointed to by PC
- c.f *locality of reference* & cache usage

Base-Register Addressing

- A holds displacement
- R holds pointer to base address
- R may be explicit or implicit
- e.g. segment registers in 80x86

Indexed Addressing

- A = base
- R = displacement
- EA = A + (R)
- Good for accessing arrays
    - EA = A + (R)
    - (R) = (R) + 1

Combinations

- Postindex
- EA = (A) + (R)

- Preindex
- EA = (A+(R))

- (Draw the diagrams)

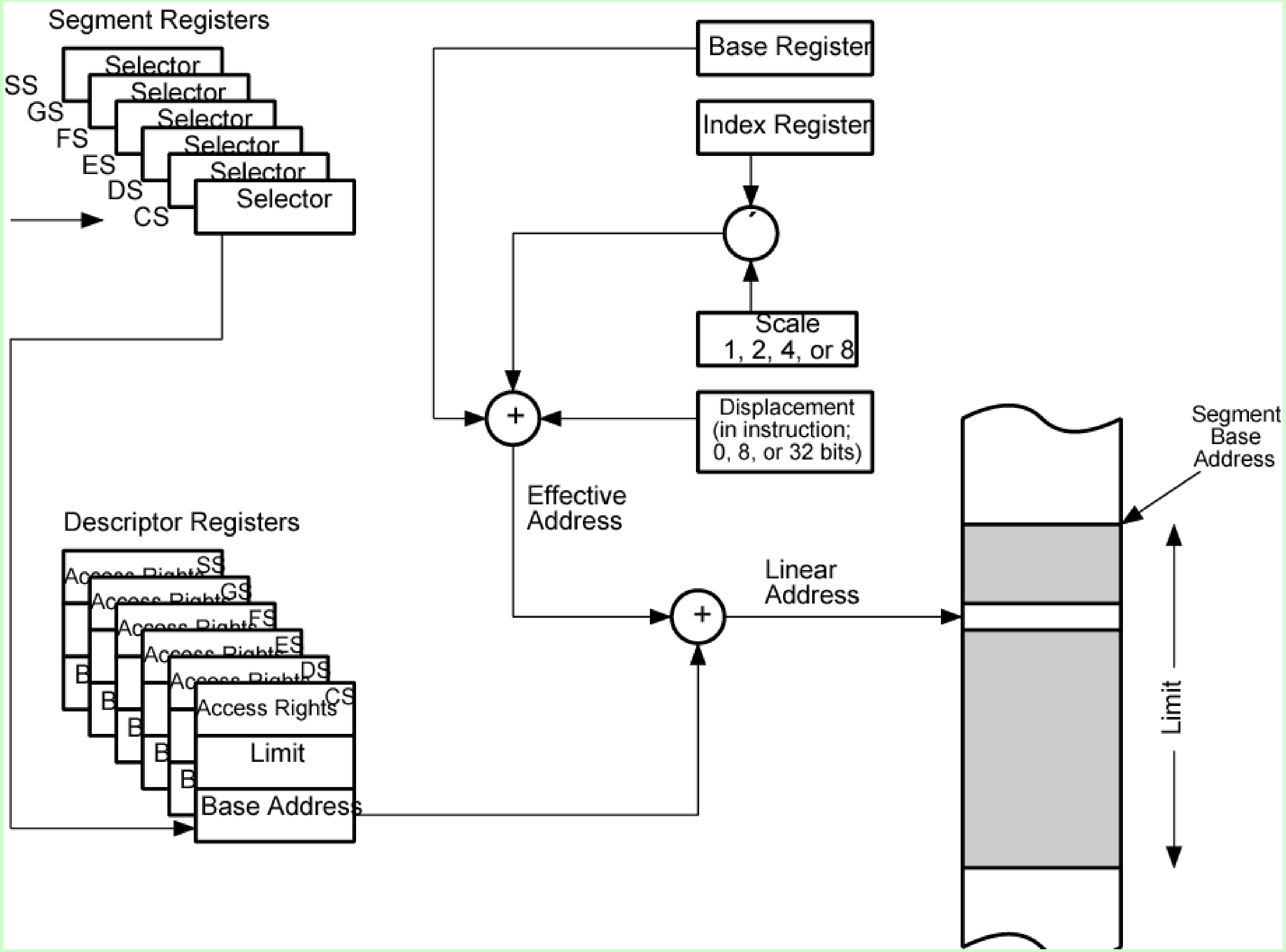
Stack Addressing

- Operand is (implicitly) on top of stack
- e.g.
  - ADD Pop top two items from stack and add

x86 Addressing Modes

- Virtual or effective address is *offset into segment*
  - Starting address plus offset gives linear address
  - This goes through page translation if paging enabled
- 12 addressing modes available
  - Immediate
  - Register operand
  - Displacement
  - Base
  - Base with displacement
  - Scaled index with displacement
  - Base with index and displacement
  - Base scaled index with displacement
  - Relative

x86 Addressing Mode Calculation



ARM Addressing Modes
Load/Store

- Only instructions that reference memory
- Indirectly through base register plus offset
- Offset
  - Offset added to or subtracted from base register contents to form the memory address
- Preindex
  - Memory address is formed as for offset addressing
  - Memory address also written back to base register
  - So base register value incremented or decremented by offset value
- Postindex
  - Memory address is base register value
  - Offset added or subtracted
    Result written back to base register
- Base register acts as index register for preindex and postindex addressing
- Offset either immediate value in instruction or another register
- If register scaled register addressing available
  - Offset register value scaled by shift operator
  - Instruction specifies shift size

Instruction Formats

- Layout of bits in an instruction
- Includes opcode
- Includes (implicit or explicit) operand(s)
- Usually *more than one instruction format* in an instruction set

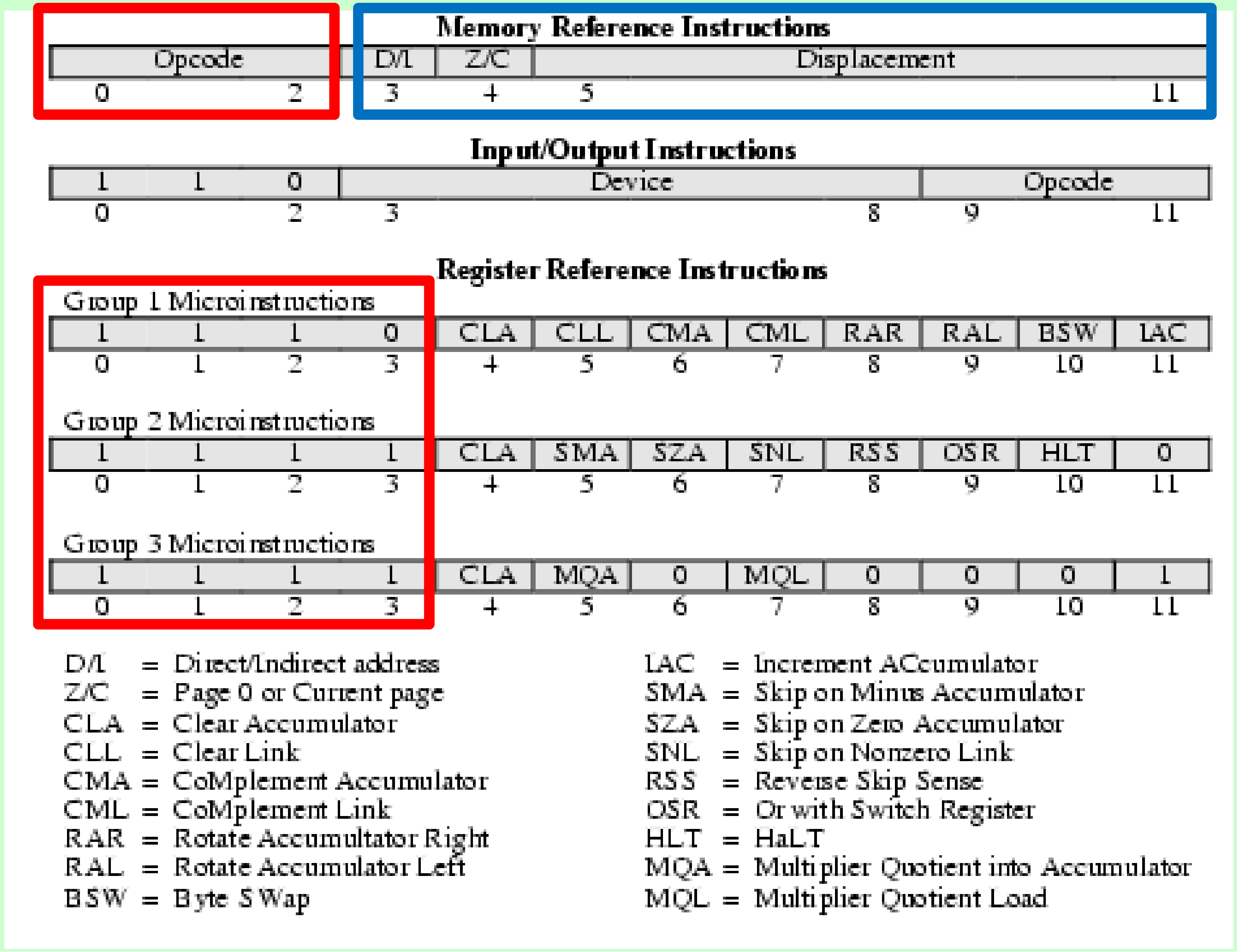
Instruction Length

- Affected by and affects:
  - Memory size
  - Memory organization
  - Bus structure
  - CPU complexity
  - CPU speed
- Trade off between powerful *instruction repertoire* and *saving space*

Allocation of Bits

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

PDP-8 Instruction Format



Memory Reference Instructions

Opcode		D/I	Z/C	Displacement	
0	2	3	4	5	11

Input/Output Instructions

1	1	0	Device		Opcode	
0		2	3	8	9	11

Register Reference Instructions

Group 1 Microinstructions

1	1	1	0	CLA	CLL	CMA	CML	RAR	RAL	BSW	IAC
0	1	2	3	4	5	6	7	8	9	10	11

Group 2 Microinstructions

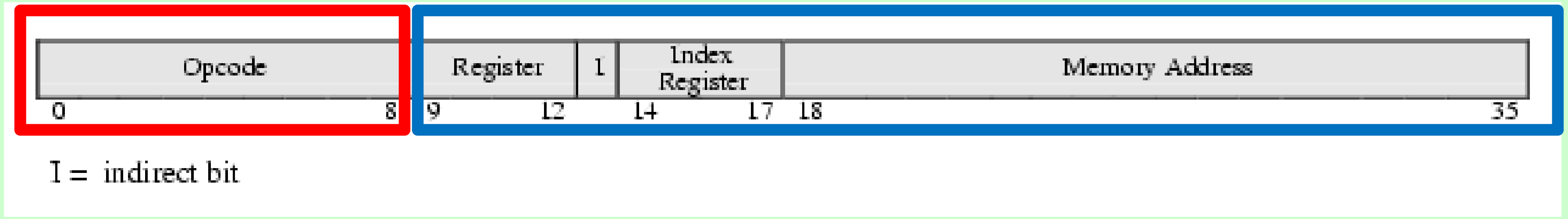
1	1	1	1	CLA	SMA	SZA	SNL	RSS	OSR	HLT	0
0	1	2	3	4	5	6	7	8	9	10	11

Group 3 Microinstructions

1	1	1	1	CLA	MQA	0	MQL	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11

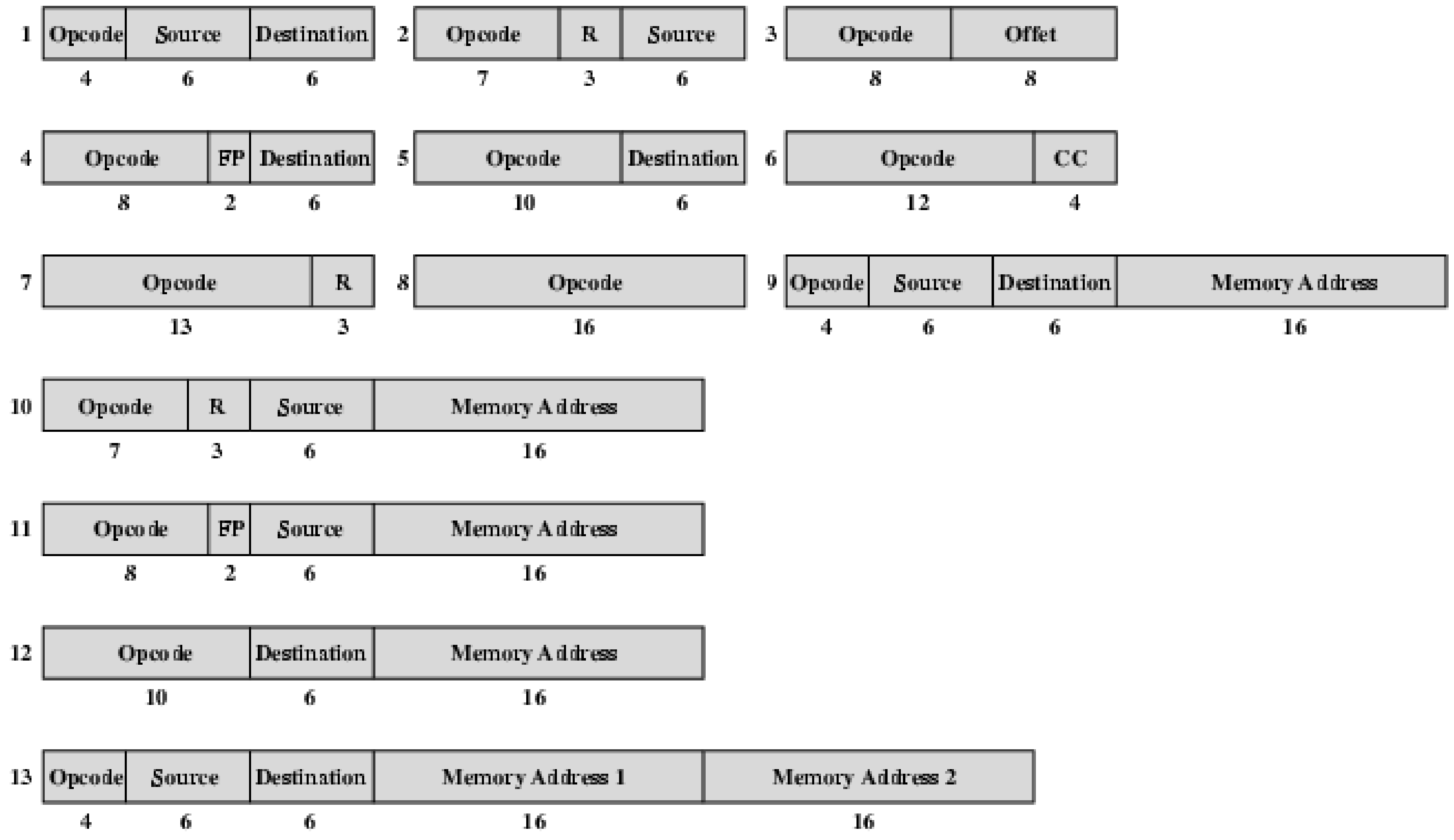
D/I	=	Direct/Indirect address	IAC	=	Increment ACcumulator
Z/C	=	Page 0 or Current page	SMA	=	Skip on Minus Accumulator
CLA	=	Clear Accumulator	SZA	=	Skip on Zero Accumulator
CLL	=	Clear Link	SNL	=	Skip on Nonzero Link
CMA	=	CoMplement Accumulator	RSS	=	Reverse Skip Sense
CML	=	CoMplement Link	OSR	=	Or with Switch Register
RAR	=	Rotate Accumultator Right	HLT	=	HaLT
RAL	=	Rotate Accumulator Left	MQA	=	Multiplier Quotient into Accumulator
BSW	=	Byte SWap	MQL	=	Multiplier Quotient Load

PDP-10 Instruction Format



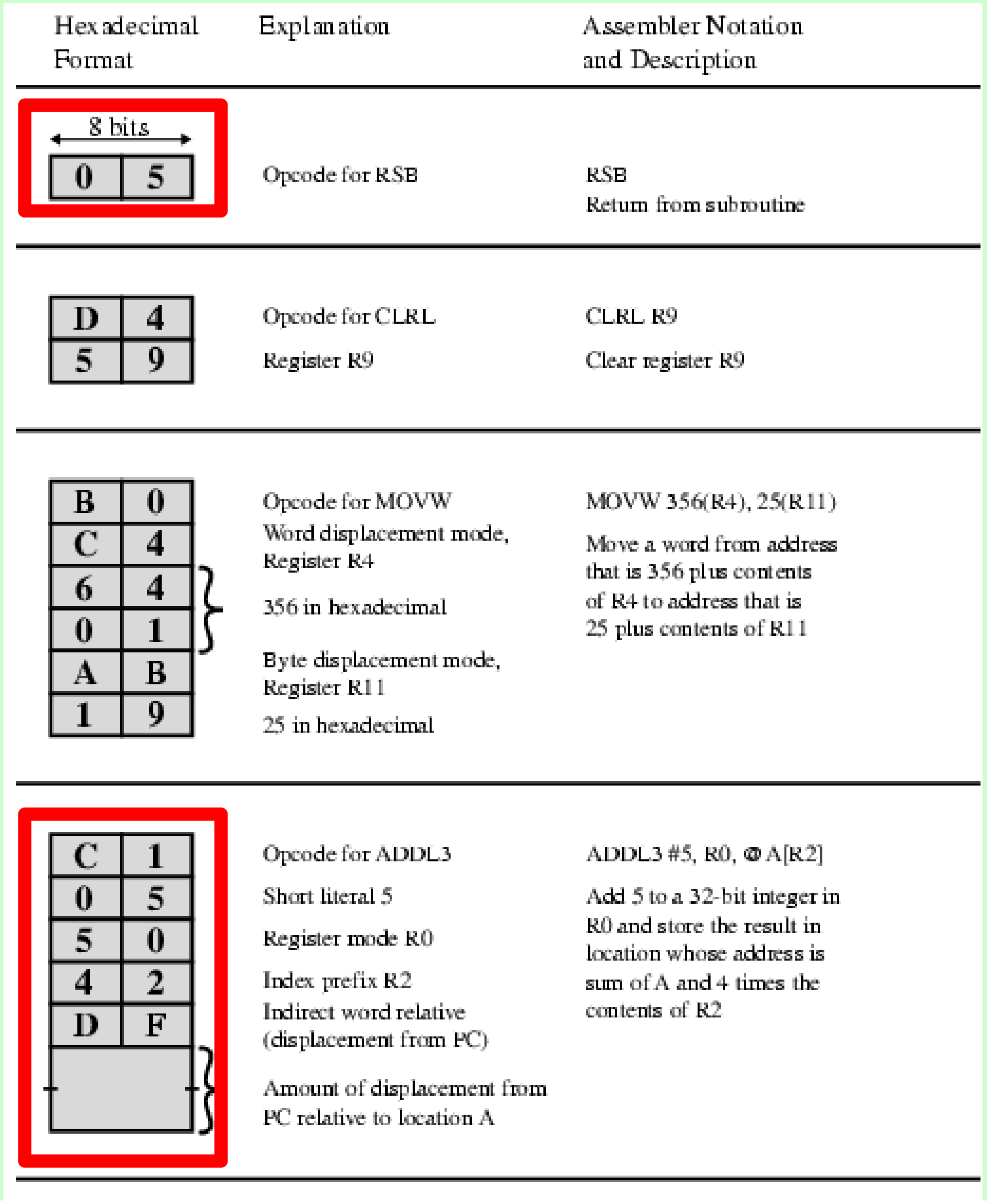
I = indirect bit

PDP-11 Instruction Format



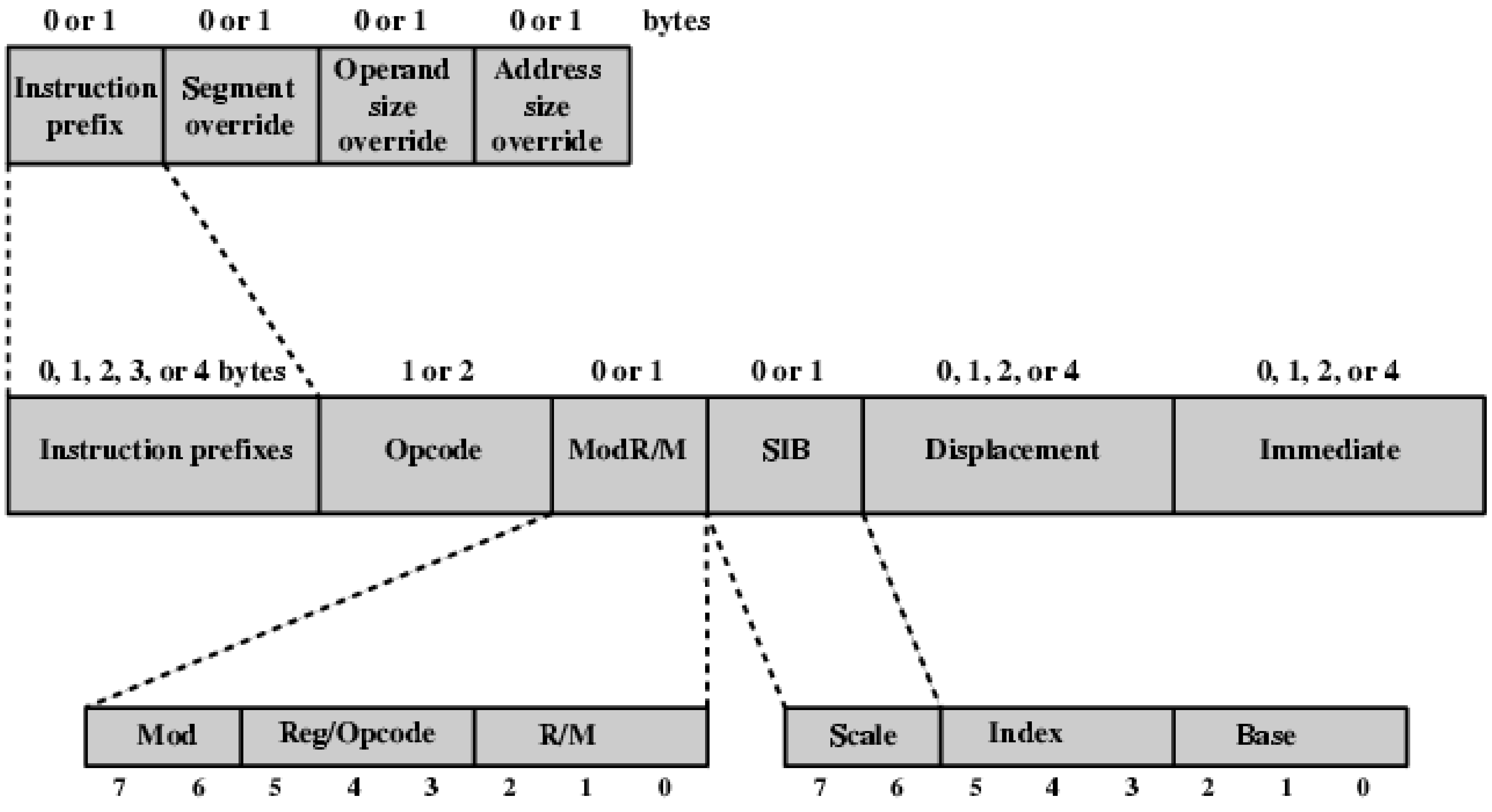
Numbers below fields indicate bit length
Source and Destination each contain a 3-bit addressing mode field and a 3-bit register number
FP indicates one of four floating-point registers
R indicates one of the general-purpose registers
CC is the condition code field

VAX Instruction Examples

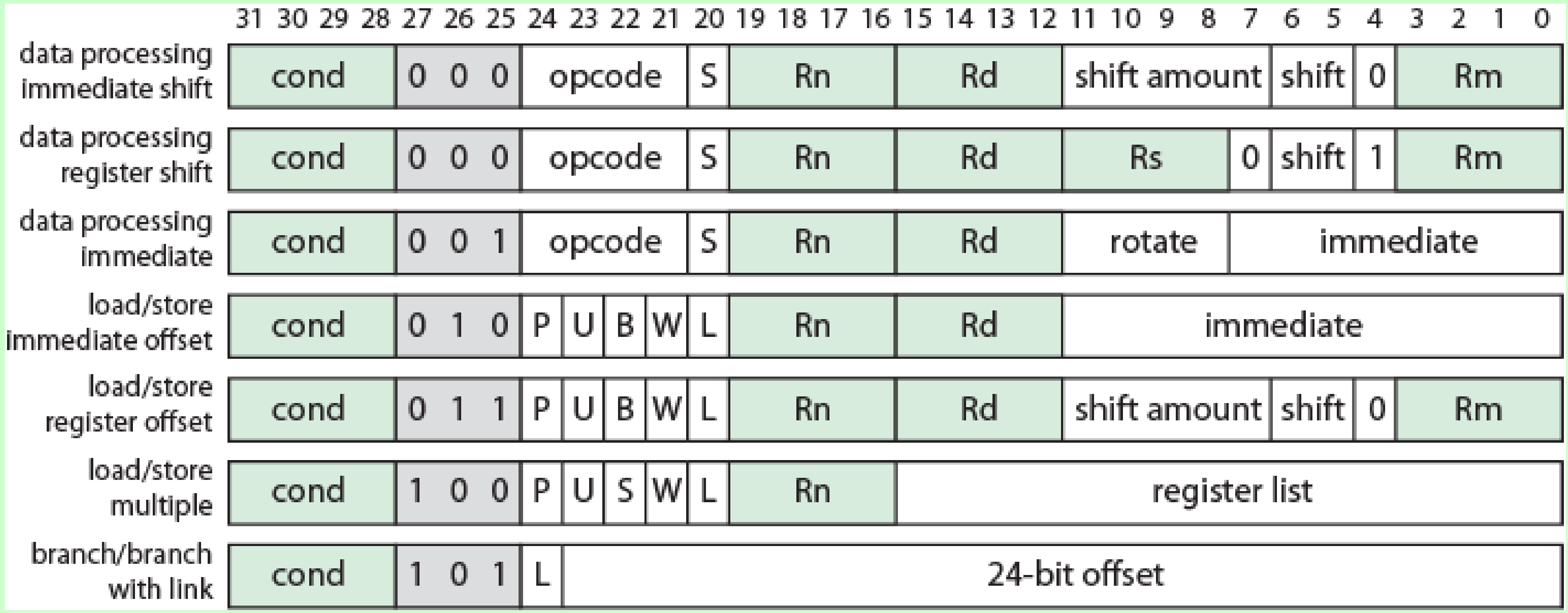


Hexadecimal Format	Explanation	Assembler Notation and Description
8 bits 0 5	Opcode for RSB	RSB Return from subroutine
D 4 5 9	Opcode for CLRL Register R9	CLRL R9 Clear register R9
B 0 C 4 6 4 0 1 A B 1 9	Opcode for MOVW Word displacement mode, Register R4 356 in hexadecimal Byte displacement mode, Register R11 25 in hexadecimal	MOVW 356(R4), 25(R11) Move a word from address that is 356 plus contents of R4 to address that is 25 plus contents of R11
C 1 0 5 5 0 4 2 D F	Opcode for ADDL3 Short literal 5 Register mode R0 Index prefix R2 Indirect word relative (displacement from PC) Amount of displacement from PC relative to location A	ADDL3 #5, R0, @ A[R2] Add 5 to a 32-bit integer in R0 and store the result in location whose address is sum of A and 4 times the contents of R2

x86 Instruction Format



ARM Instruction Formats



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
data processing immediate shift	cond				0	0	0	opcode				S	Rn				Rd				shift amount					shift		0	Rm			
data processing register shift	cond				0	0	0	opcode				S	Rn				Rd				Rs				0	shift		1	Rm			
data processing immediate	cond				0	0	1	opcode				S	Rn				Rd				rotate				immediate							
load/store immediate offset	cond				0	1	0	P	U	B	W	L	Rn				Rd				immediate											
load/store register offset	cond				0	1	1	P	U	B	W	L	Rn				Rd				shift amount					shift		0	Rm			
load/store multiple	cond				1	0	0	P	U	S	W	L	Rn				register list															
branch/branch with link	cond				1	0	1	L	24-bit offset																							

- S = For data processing instructions, updates condition codes
- S = For load/store multiple instructions, execution restricted to supervisor mode
- P, U, W = distinguish between different types of addressing_mode
- B = Unsigned byte (B==1) or word (B==0) access
- L = For load/store instructions, Load (L==1) or Store (L==0)
- L = For branch instructions, is return address stored in link register

Homework

- Reading book
- 1.Translate Key Terms
- 2.Problems
  - 11.2
  - 11.3
  - 11.4
  - 11.5
  - 11.6