William Stallings
Computer Organization and Architecture
7th Edition

Chapter 11

Instruction Sets:

Addressing Modes and Formats

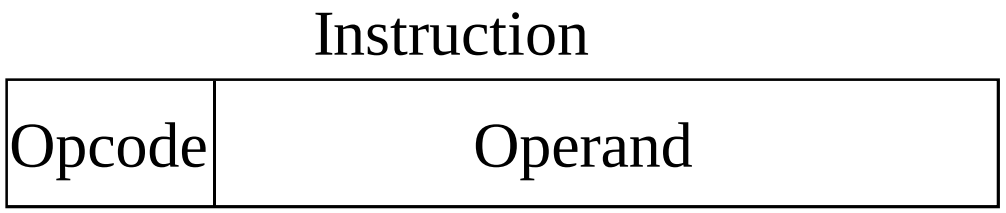
Addressing Modes

- 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
  - 5 is operand
- No memory reference to fetch data
- Fast
- Limited range

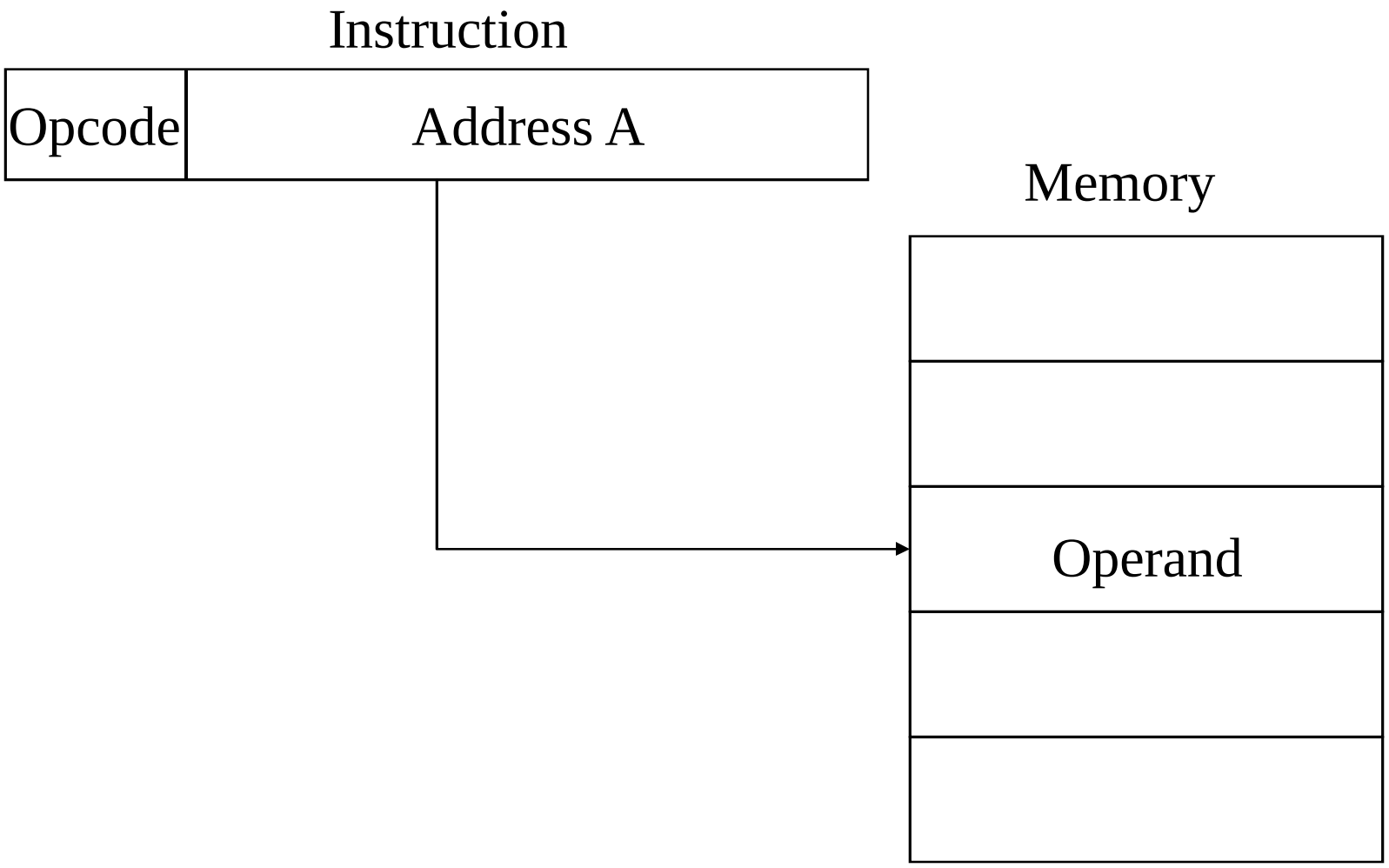
Immediate Addressing Diagram



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



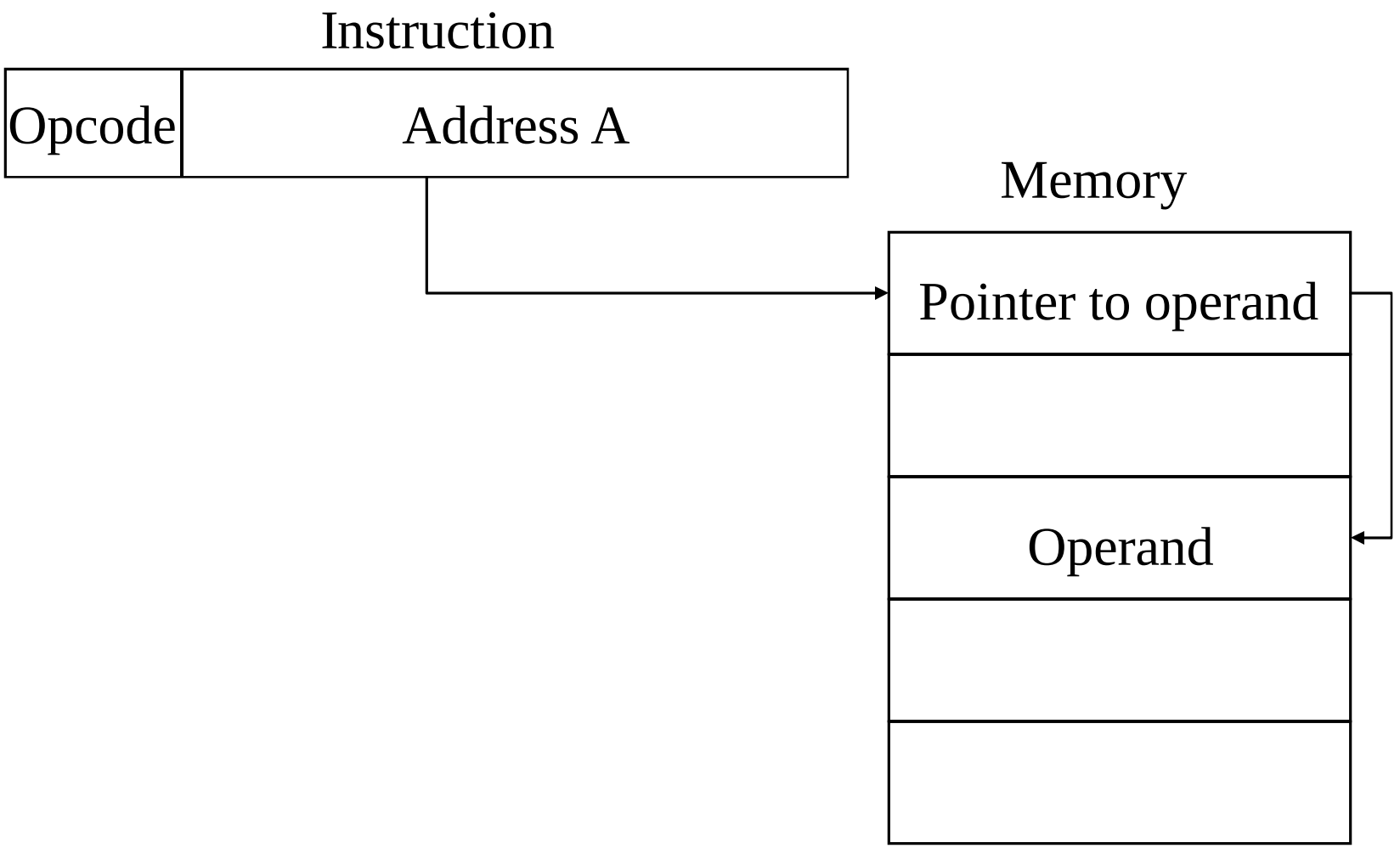
Indirect Addressing (1)

- Memory cell pointed to by address field contains the address of (pointer to) 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 (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
- Hence slower

Indirect Addressing Diagram



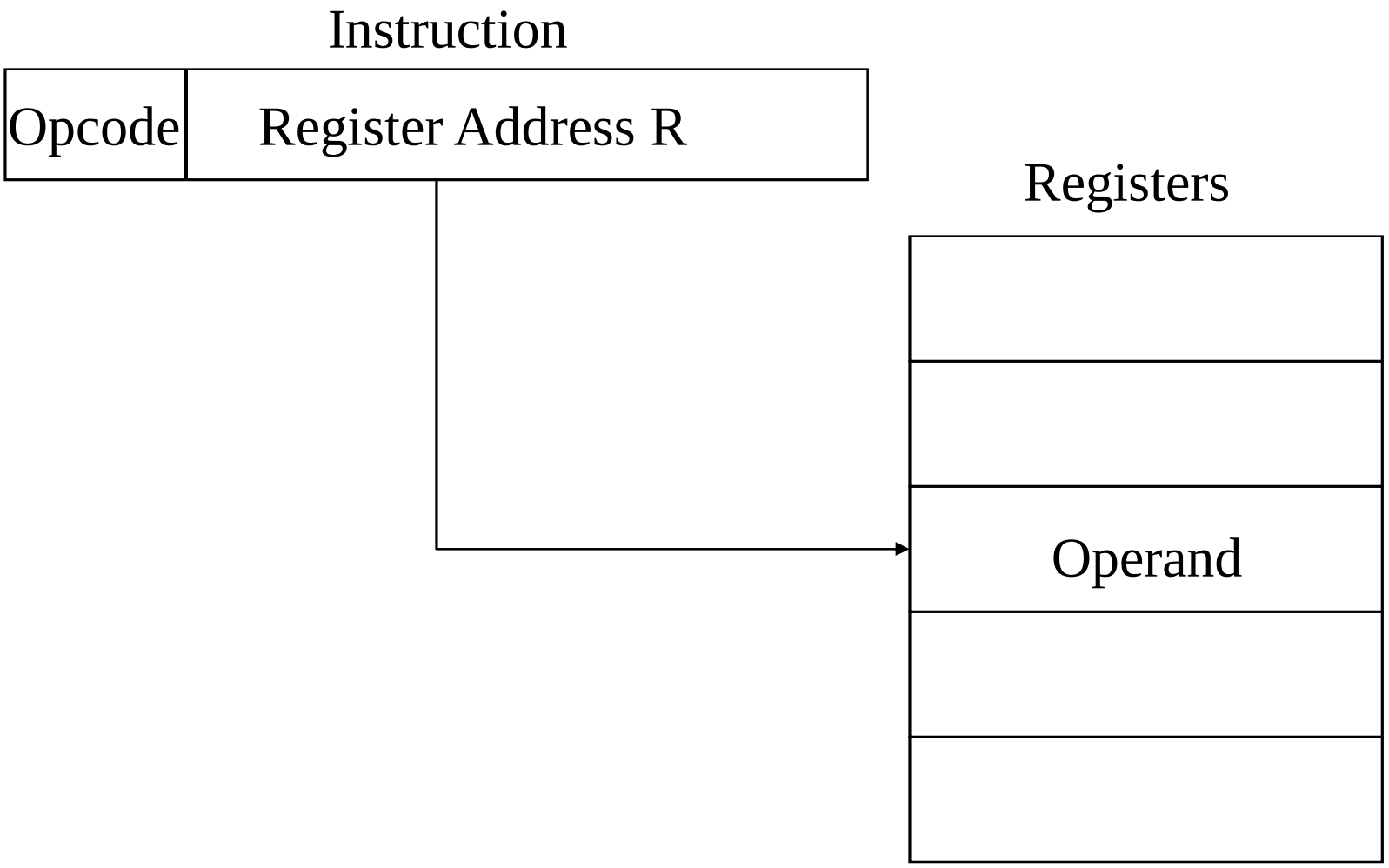
Register Addressing (1)

- Operand is held in register named in address filed
- EA = R
- Limited number of registers
- Very small address field needed
  - —Shorter instructions
  - —Faster instruction fetch

Register Addressing (2)

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

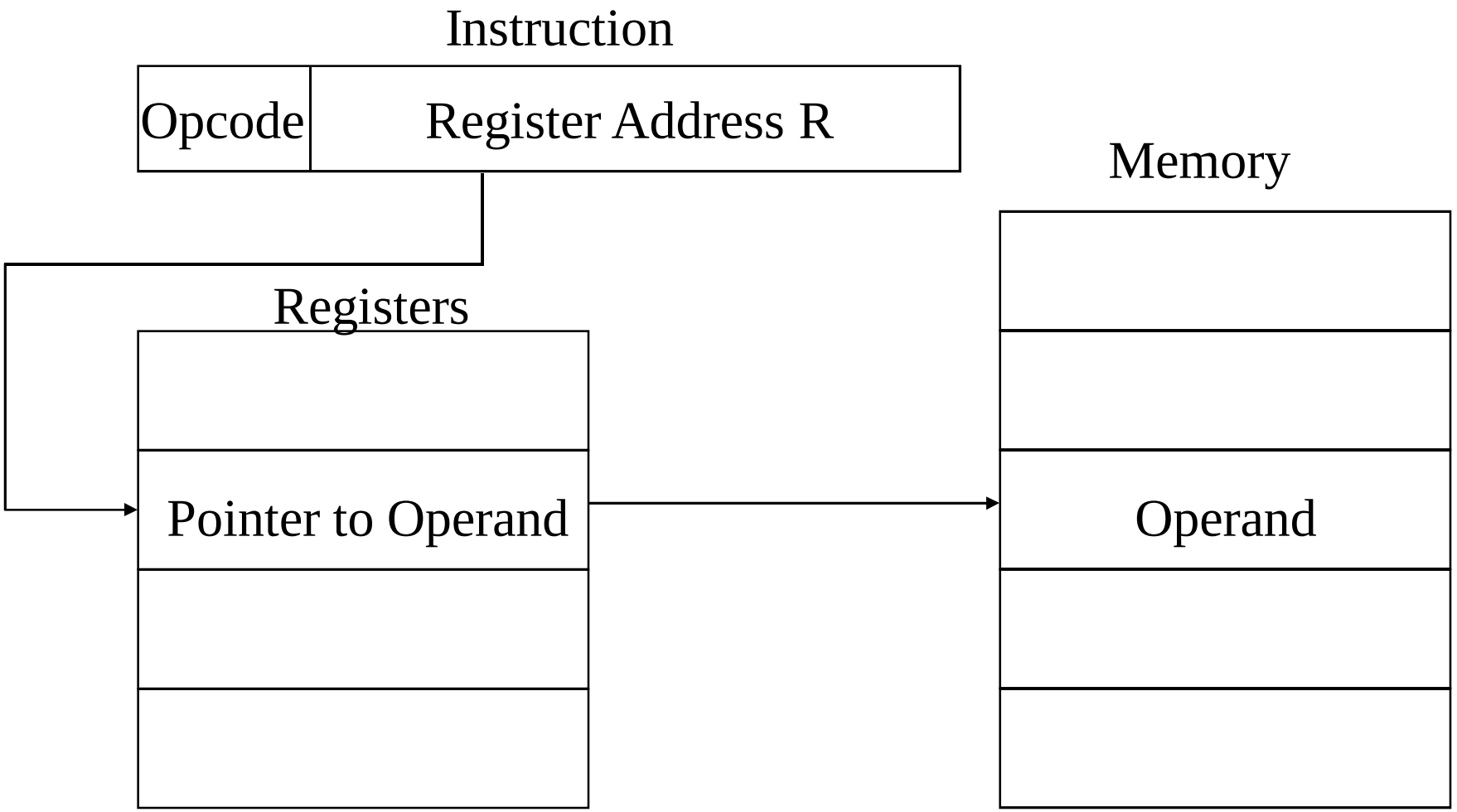
Register Addressing Diagram



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

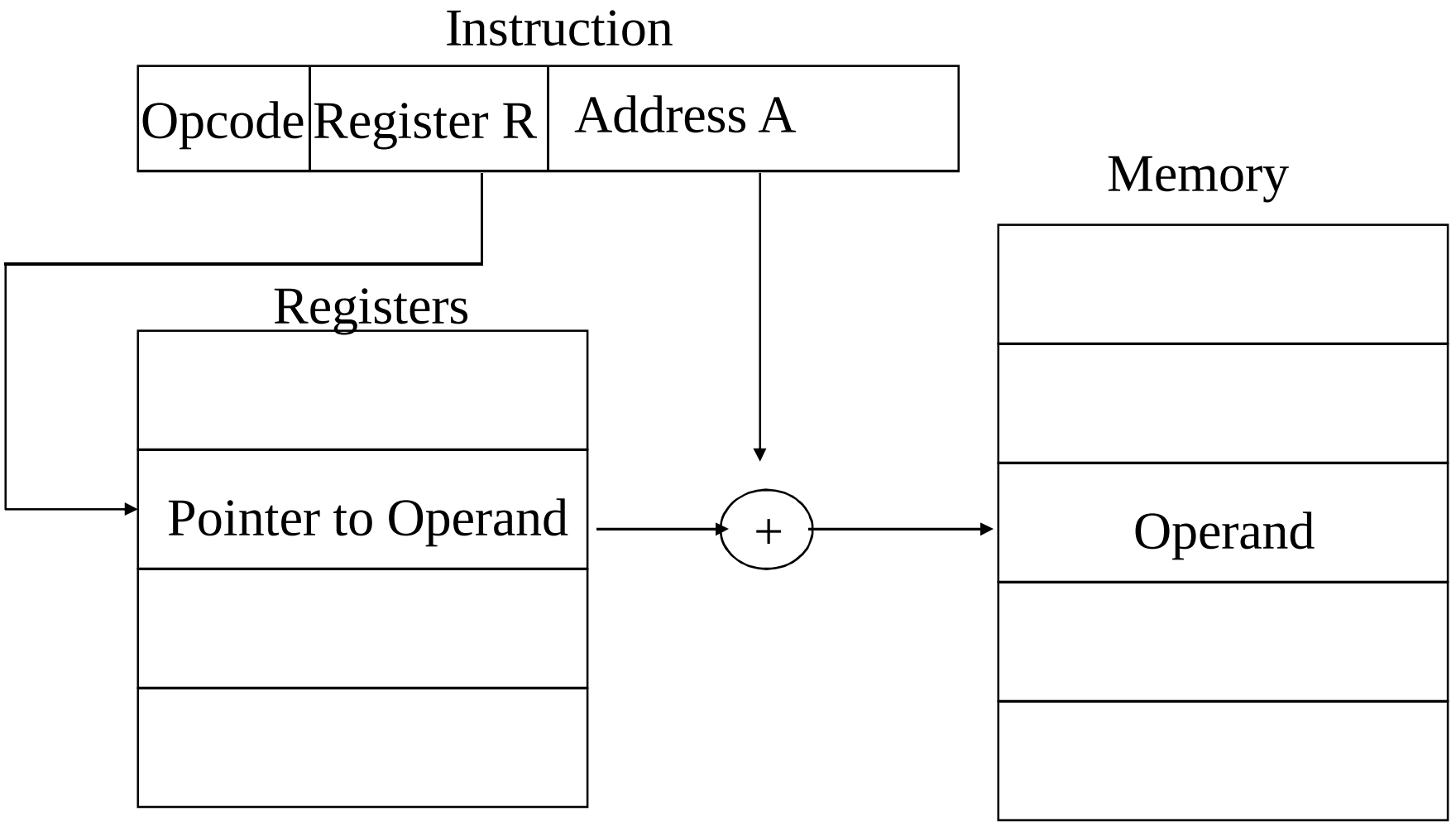
Register Indirect Addressing Diagram



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++

Combinations

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

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

- **(Draw the diagrams)**

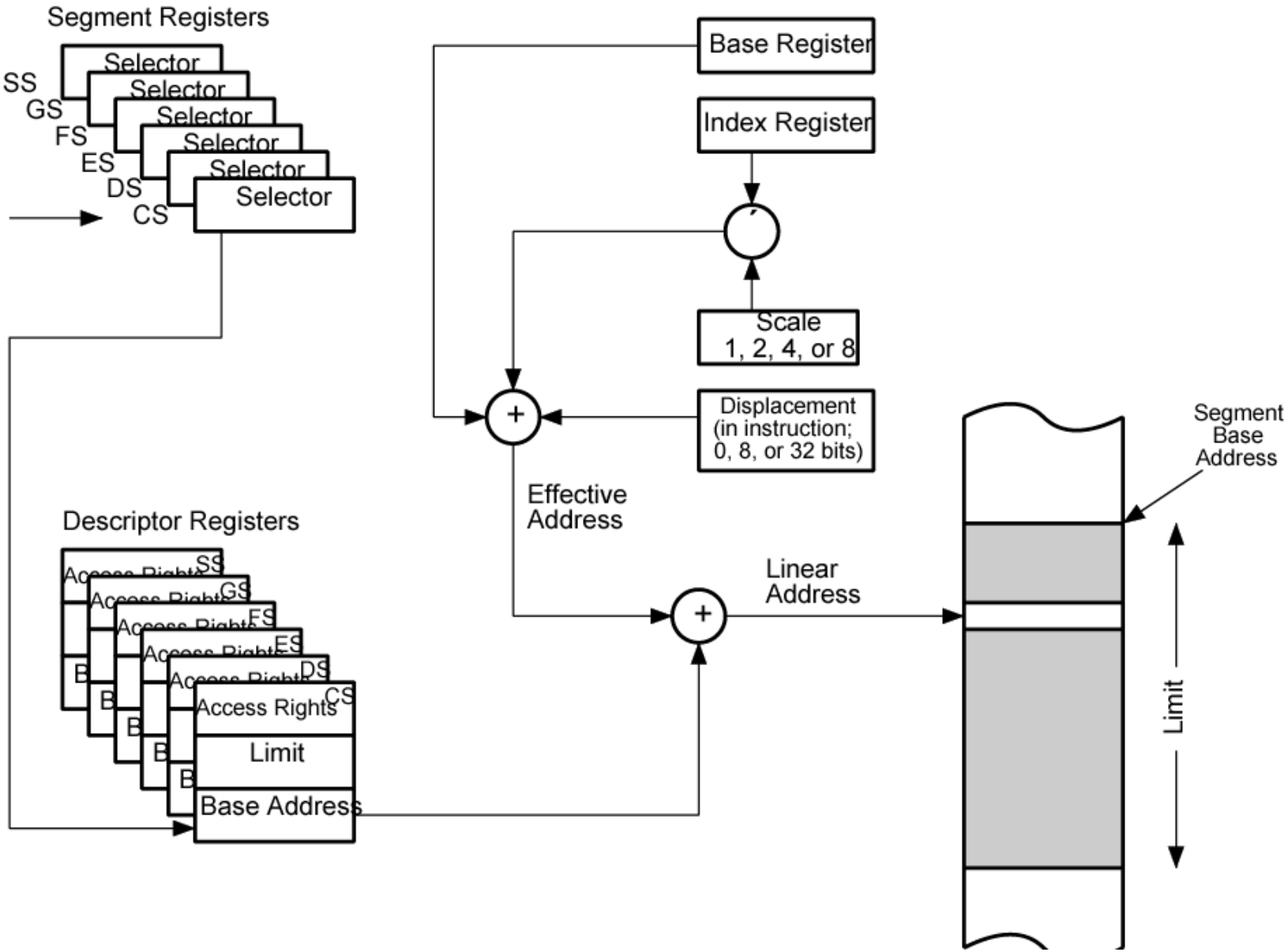
Stack Addressing

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

Pentium 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

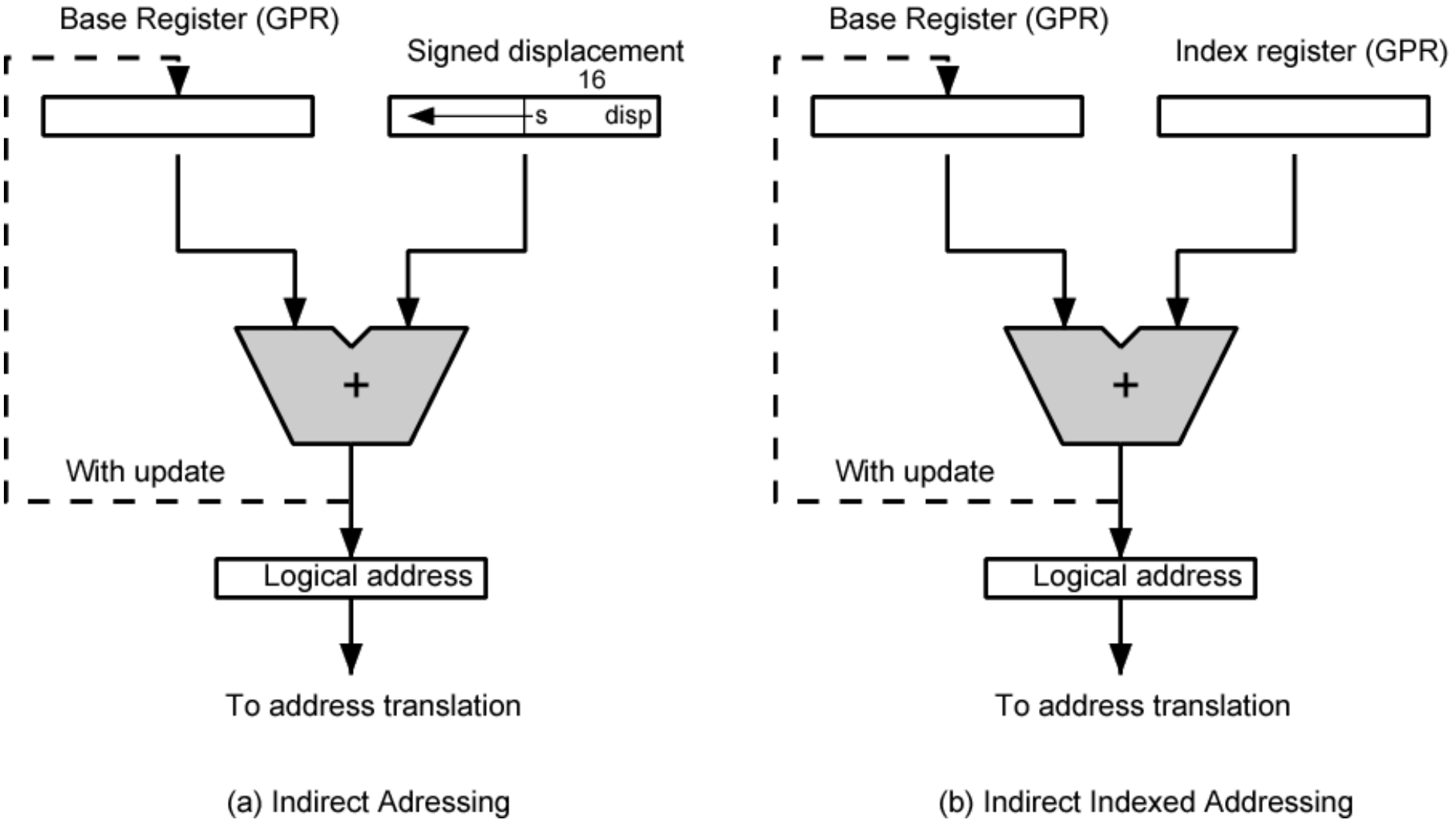
Pentium Addressing Mode Calculation



PowerPC Addressing Modes

- Load/store architecture
  - Indirect
    - Instruction includes 16 bit displacement to be added to base register (may be GP register)
    - Can replace base register content with new address
  - Indirect indexed
    - Instruction references base register and index register (both may be GP)
    - EA is sum of contents
- Branch address
  - Absolute
  - Relative
  - Indirect
- Arithmetic
  - Operands in registers or part of instruction
  - Floating point is register only

PowerPC Memory Operand Addressing Modes



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

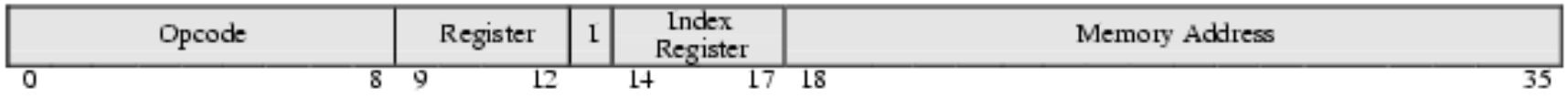
Group 3 Microinstructions

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
Z/C = Page 0 or Current page
CLA = Clear Accumulator
CLL = Clear Link
CMA = CoMplement Accumulator
CML = CoMplement Link
RAR = Rotate Accumultator Right
RAL = Rotate Accumulator Left
BSW = Byte SWap

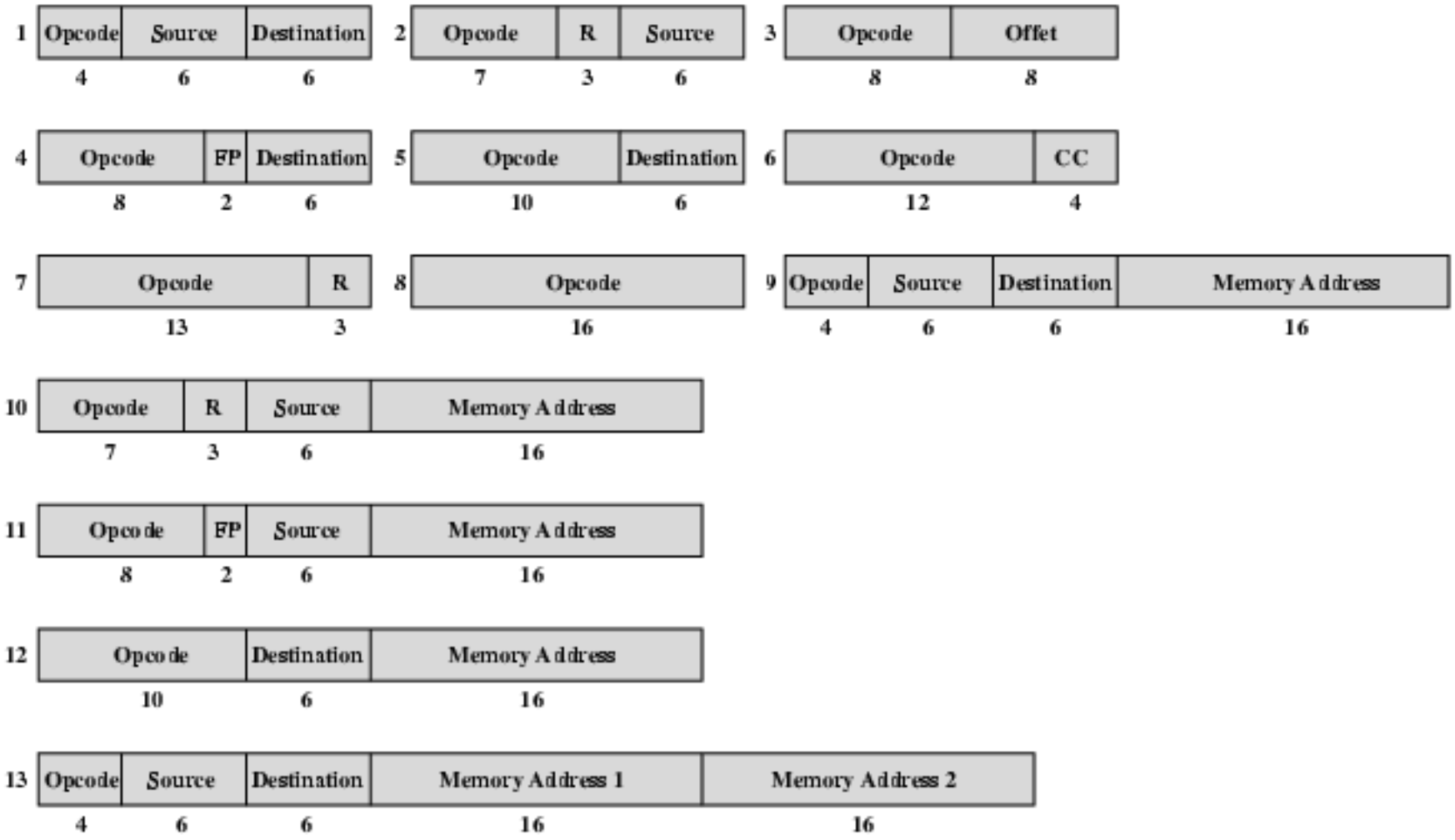
IAC = Increment ACcumulator
SMA = Skip on Minus Accumulator
SZA = Skip on Zero Accumulator
SNL = Skip on Nonzero Link
RSS = Reverse Skip Sense
OSR = Or with Switch Register
HLT = HaLT
MQA = Multiplier Quotient into Accumulator
MQL = Multiplier Quotient Load

PDP-10 Instruction Format



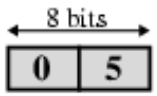
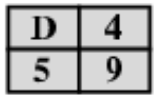
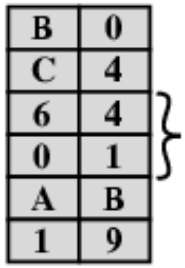
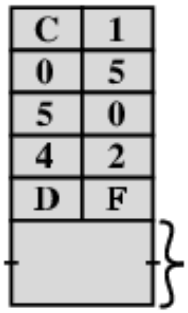
I = indirect bit

PDP-11 Instruction Format

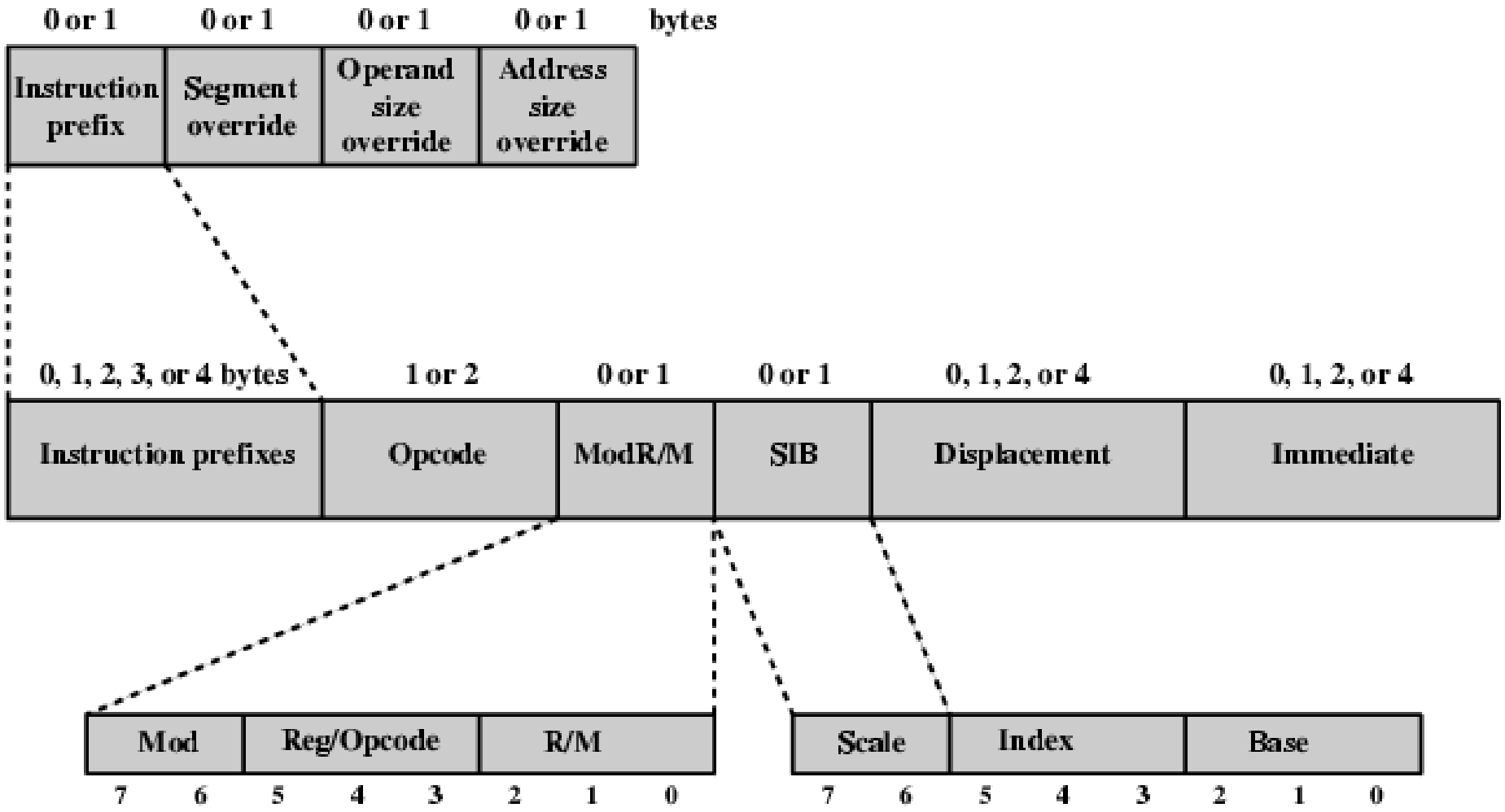


Numbers below fields indicate bit length
Source and Destination each contain a 3-bit addressing mode field and a 3-bit register number
FP indicates one of four floating-point registers
R indicates one of the general-purpose registers
CC is the condition code field

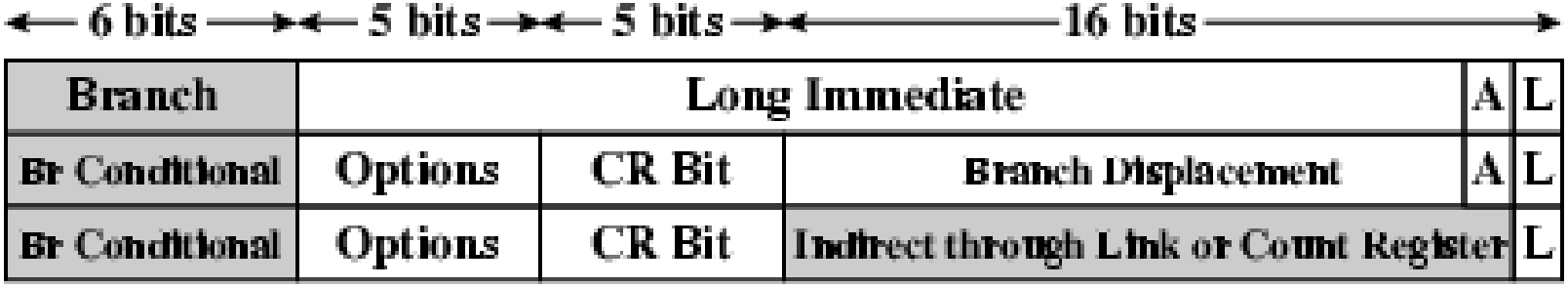
VAX Instruction Examples

Hexadecimal Format	Explanation	Assembler Notation and Description
	Opcode for RSB	RSB Return from subroutine
	Opcode for CLRL Register R9	CLRL R9 Clear register R9
	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
	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

Pentium Instruction Format



PowerPC Instruction Formats (1)



(a) Branch instructions

CR	Dest Bit	Source Bit	Source Bit	Add, OR, XOR, etc.	/

(b) Condition register logical instructions

Ld/St Indirect	Dest Register	Base Register	Displacement		
Ld/St Indirect	Dest Register	Base Register	Index Register	Size, Sign, Update	/
Ld/St Indirect	Dest Register	Base Register	Displacement		XO

(c) Load/store instructions

PowerPC Instruction Formats (2)

Ld/St Indirect	Dest Register	Base Register	Displacement				
Ld/St Indirect	Dest Register	Base Register	Index Register	Size, Sign, Update		/	
Ld/St Indirect	Dest Register	Base Register	Displacement			XO	*

(c) Load/store instructions

Arithmetic	Dest Register	Src Register	Src Register	O	Add, Sub, etc.		R	
Add, Sub, etc.	Dest Register	Src Register	Signed Immediate Value					
Logical	Src Register	Dest Register	Src Register	ADD, OR, XOR, etc.			R	
AND, OR, etc.	Src Register	Dest Register	Unsigned Immediate Value					
Rotate	Src Register	Dest Register	Shift Amt	Mask Begin	Mask End		R	
Rotate or Shift	Src Register	Dest Register	Src Register	Shift Type or Mask			R	
Rotate	Src Register	Dest Register	Shift Amt	Mask	XO	S	R	*
Rotate	Src Register	Dest Register	Src Register	Mask	XO		R	*
Shift	Src Register	Dest Register	Shift Type or Mask			S	R	*

(d) Integer arithmetic, logical, and shift/rotate instructions

Flt sgl/dbl	Dest Register	Src Register	Src Register	Src Register	Fadd, etc.	R

(e) Floating-point arithmetic instructions

A = Absolute or PC relative
L = Link or subroutine
O = Record overflow in XER
R = Record condition in CR1

XO = Opcode extension
S = Part of shift amount field
* = 64-bit implementation only

Foreground Reading

- **Stallings chapter 11**
- **Intel and PowerPC Web sites**