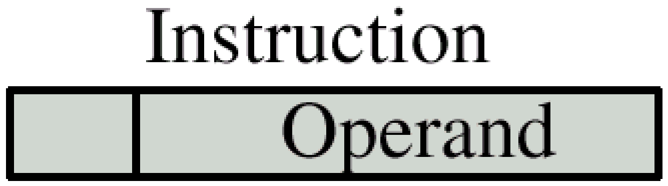
Chapter 11
Instruction Sets:
Addressing Modes and Formats

Contents

11.1 Addressing

- How to specify the locations of operands?
  - Immediate
  - Direct
  - Indirect
  - Register
  - Register Indirect
  - Displacement (Indexed)
  - Stack

Immediate Addressing

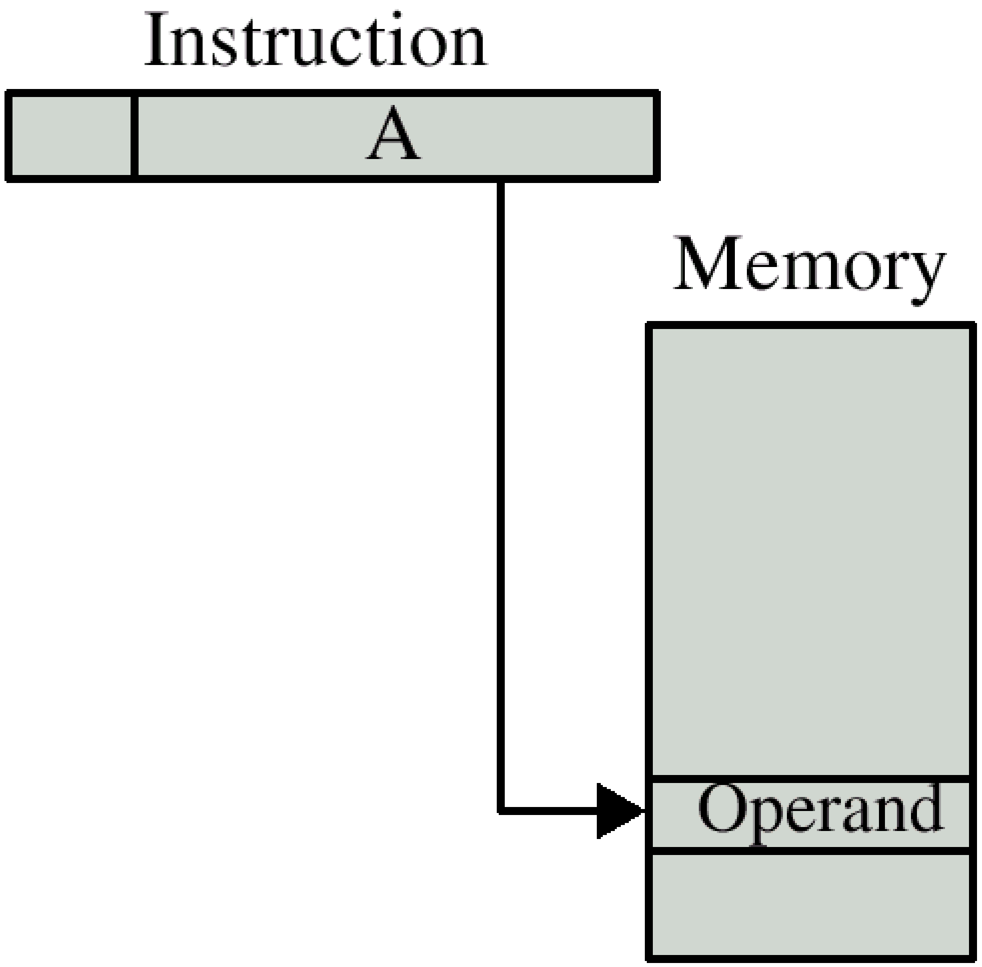


- Operand is part of instruction
  - ADD 5
    - Add 5 to contents of accumulator
    - 5 is operand
- No memory reference to fetch data
- Size of the number is restricted to the size of the address field

Direct Addressing

- Address field contains the address of operand
  - EA = A
    - Effective address is equal to the address field
- 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 get effective address
- Address space is limited to the size of the address field

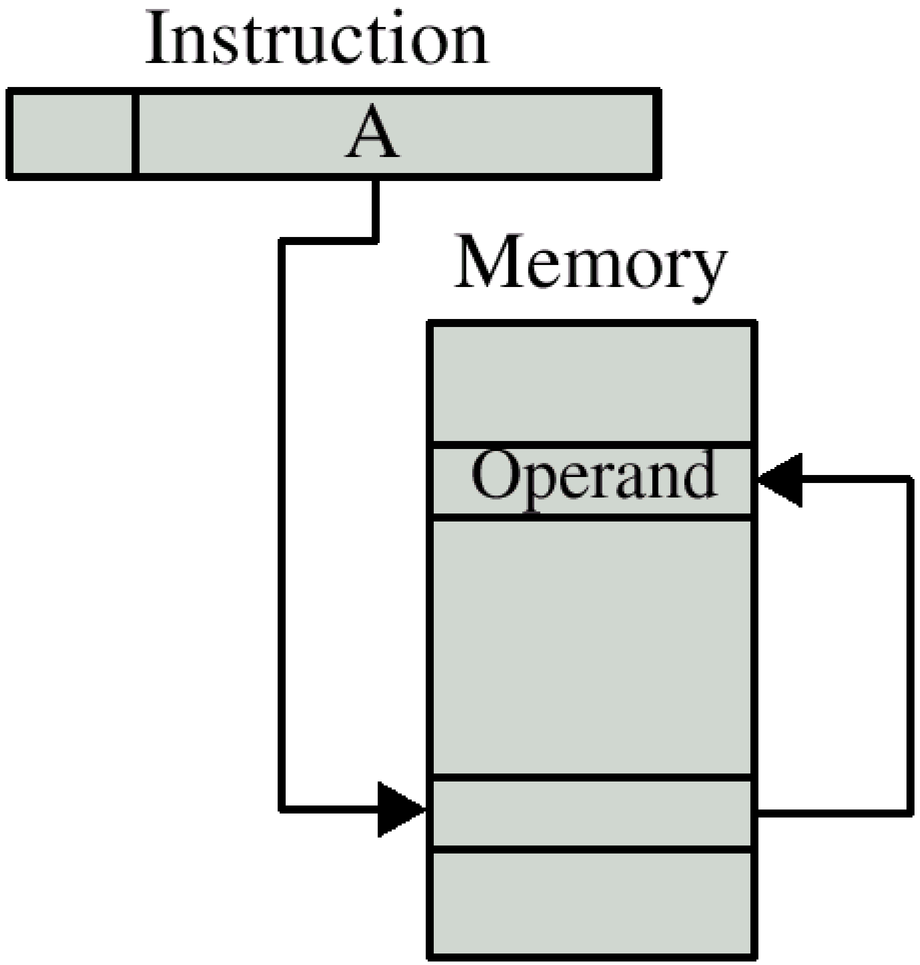
Direct Addressing



Indirect Addressing

- Memory cell pointed to by address field contains the address of the operand
- EA = (A)
  - Look in A, find address (A) and look there for operand
- ADD (A)
  - Add contents of cell pointed to by contents of A to accumulator
- Larger address space is possible
  - 2^n where n = word length
- May be nested, multilevel, cascaded
  - EA = (((A)))
- Multiple memory accesses to find operand

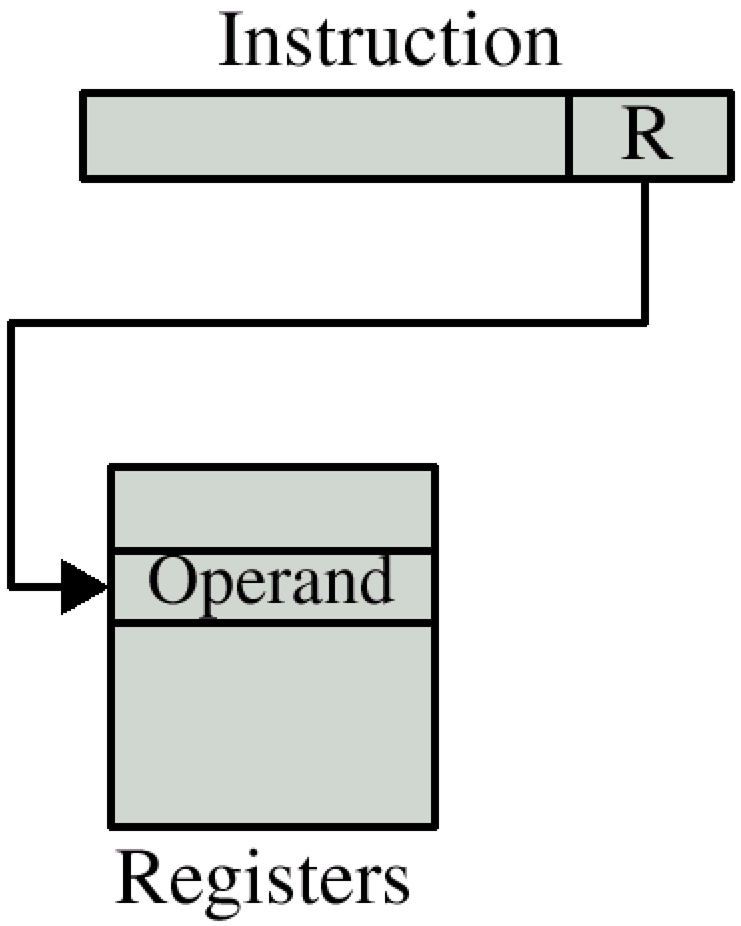
Indirect Addressing



Register Addressing

- Operand is held in register
- EA = R
- Advantages
  - Only a small address field is needed
    - Shorter instructions
  - No memory access
    - Fast execution is possible
- Very limited address space

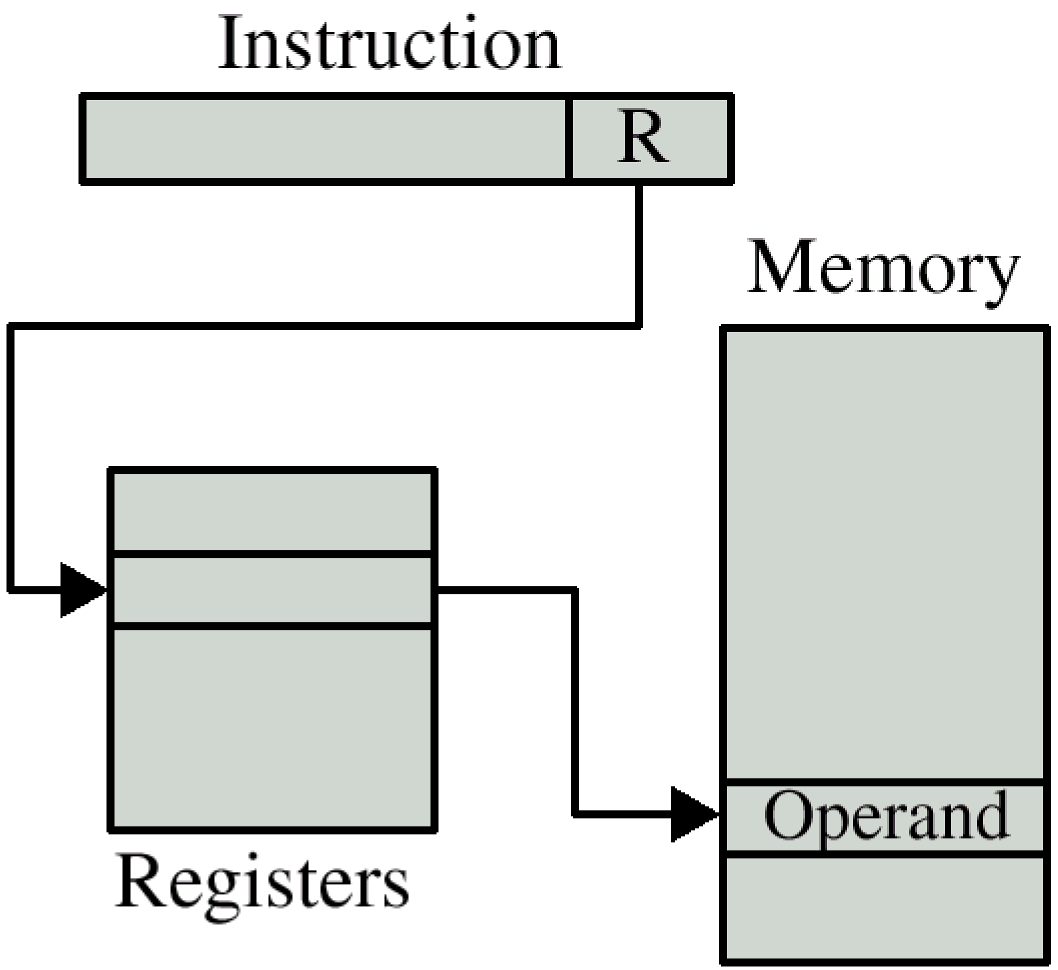
Register Addressing



Register Indirect Addressing

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

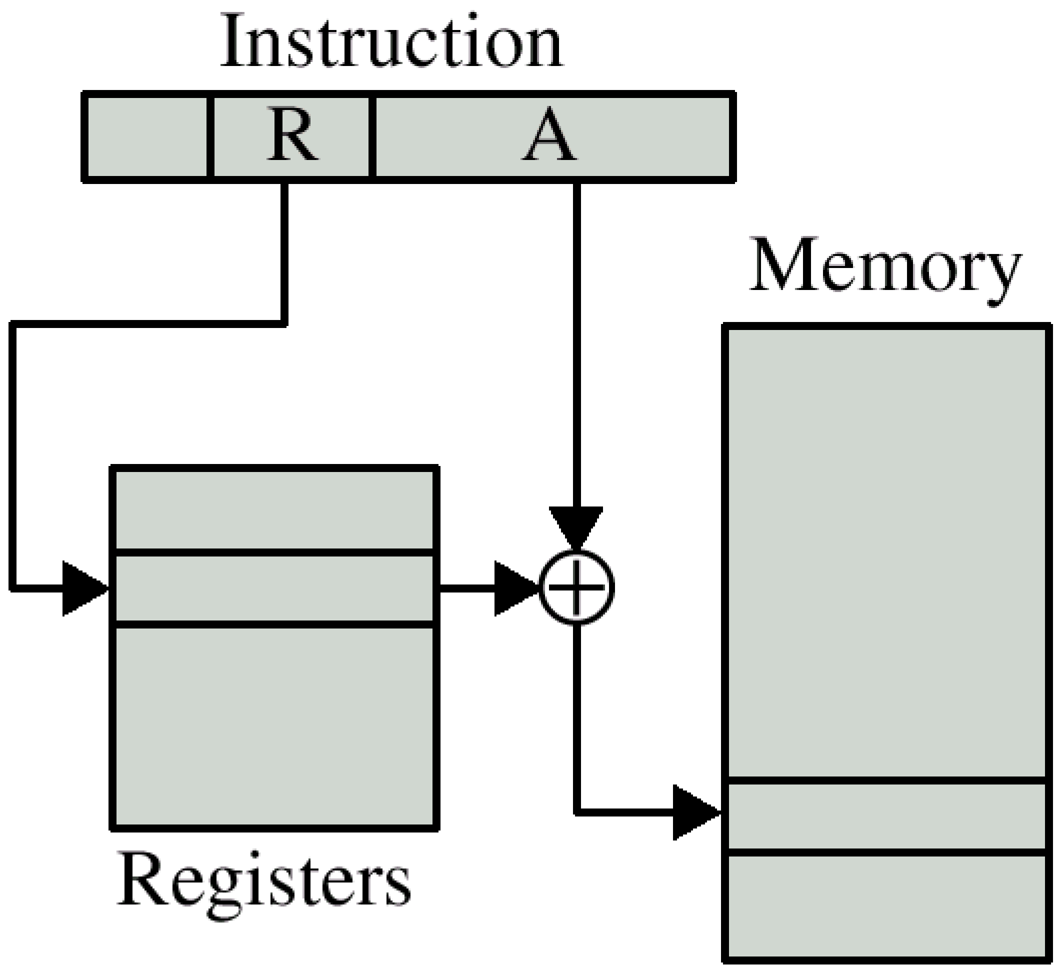
Register Indirect Addressing



Displacement Addressing

- Combines the capabilities of direct addressing and register indirect addressing
- EA = A + (R)
- Instruction has two address fields
  - A = base value
  - R = register that holds displacement
  - or vice versa
- Common uses of displacement addressing
  - Relative addressing
  - Base-register addressing
  - Indexing

Displacement Addressing



Relative Addressing

- R = PC(Program Counter)
- EA = A + (PC)
- Current instruction address is added to the address field to produce the EA
  - Address field is treated as a twos complement number for this operation
  - Effective address is a displacement relative to the address of the instruction
- LOOP: ADD BX, AX

```
LOOP:  ADD BX, AX
       .....
       JNZ  LOOP
```

Base-Register Addressing

- EA = A + (R)
- A holds displacement
- R holds pointer to base address
  - R may be explicit or implicit
- Segment registers in 80x86
  - Offset address is added to the contents of segment register

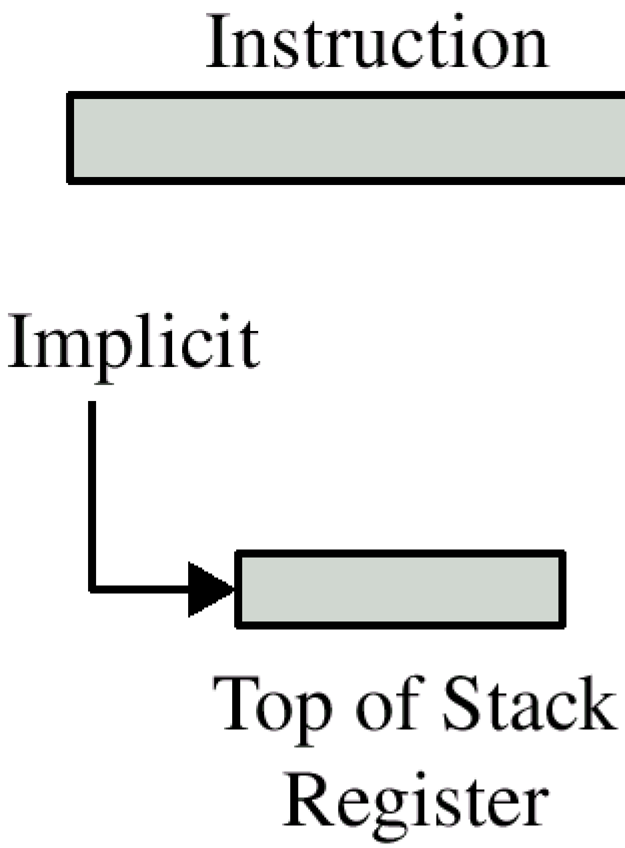
Indexing

- EA = A + (R)
- A = base
- R = positive displacement
- Opposite of base-register addressing
- Good for accessing arrays
  - EA = A + (R)
  - (R) <-- (R) + 1

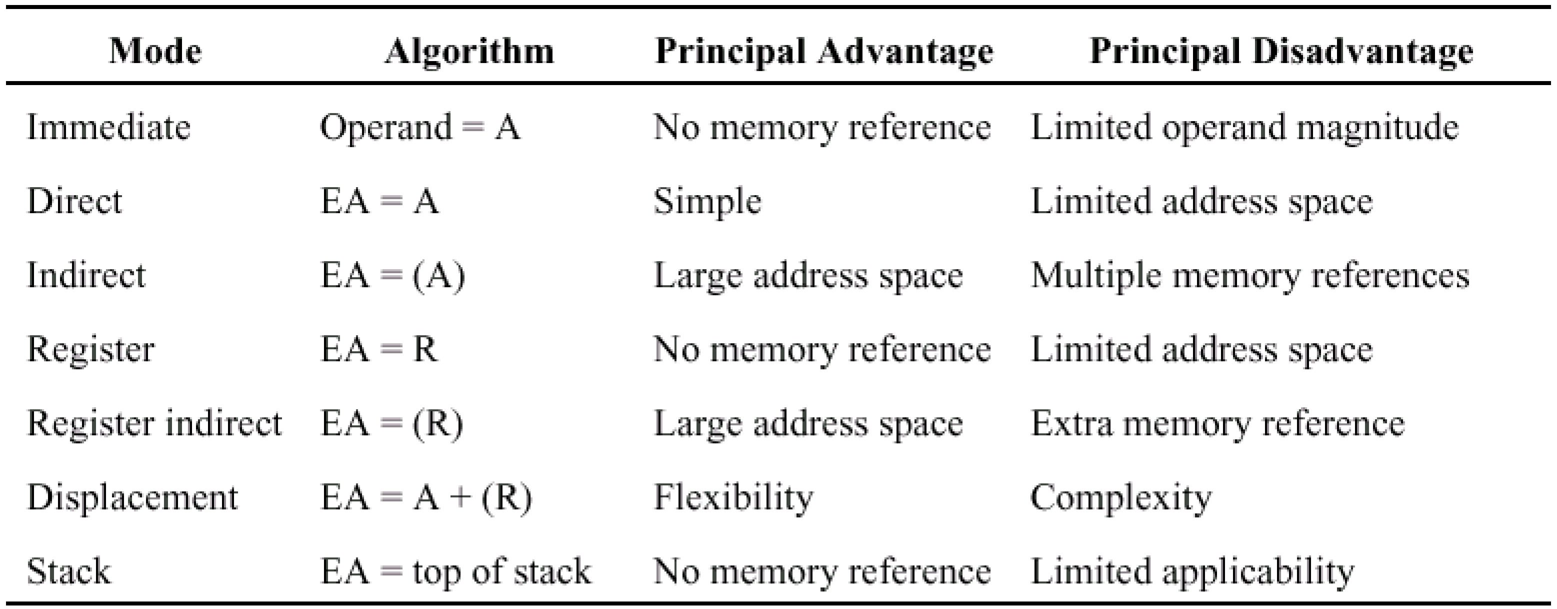
Stack Addressing

- Operand is (implicitly) on top of stack
- ADD
  - Pop top two items from stack, add, push the result onto stack

Stack Addressing



Summary of 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

11.2 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 is enabled
- 9 addressing modes available
  - Immediate
  - Register operand
  - Displacement : offset in a segment
  - Base : same as register indirect addressing
  - Base with displacement
  - Scaled index with displacement : scale factor is used
    - scale factor of 2 can be used to index an array of 16-bit integers
  - Base with index and displacement
  - Base scaled index with displacement
  - Relative
    - used in transfer-of-control instructions

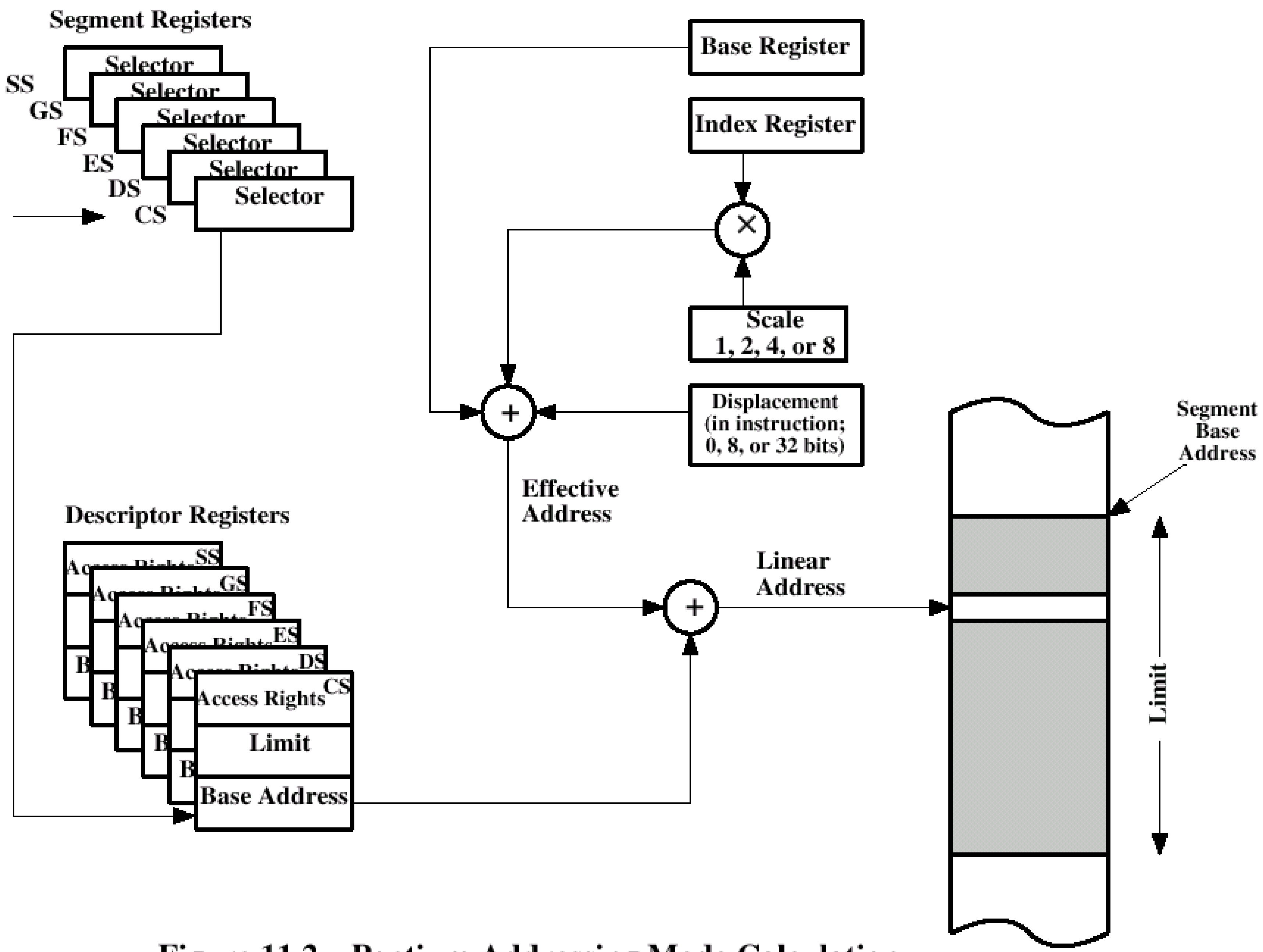


Figure 11.2 Pentium Addressing Mode Calculation

Pentium II 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

PowerPC Addressing Modes

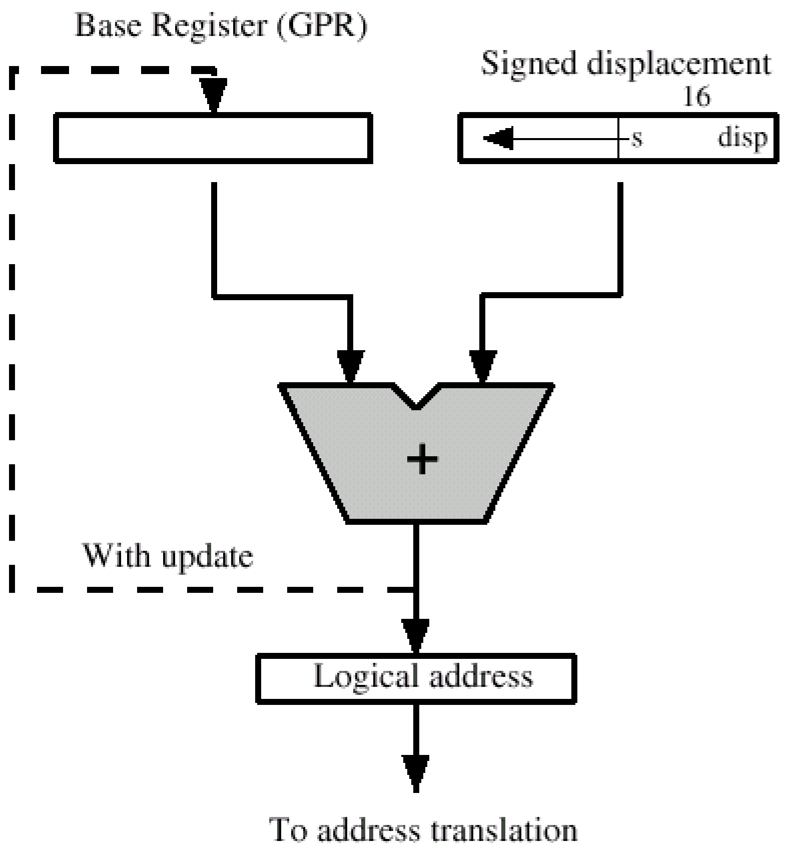
- Uses a simple set of addressing modes
  - Load/store addressing
    - Indirect : EA = (BR) + D
      - Instruction includes 16 bit displacement to be added to base register
      - Can replace base register content with new address
    - Indirect indexed : EA = (BR) + (IR)
      - Instruction references base register and index register
      - EA is sum of contents of both registers
  - Branch addressing
    - Absolute : EA = I
    - Relative : EA = (PC) + I
    - Indirect : EA = (L/CR)
  - Arithmetic instructions
    - Operands in registers or part of instruction
    - Register addressing only for floating point computation

PowerPC Addressing Modes

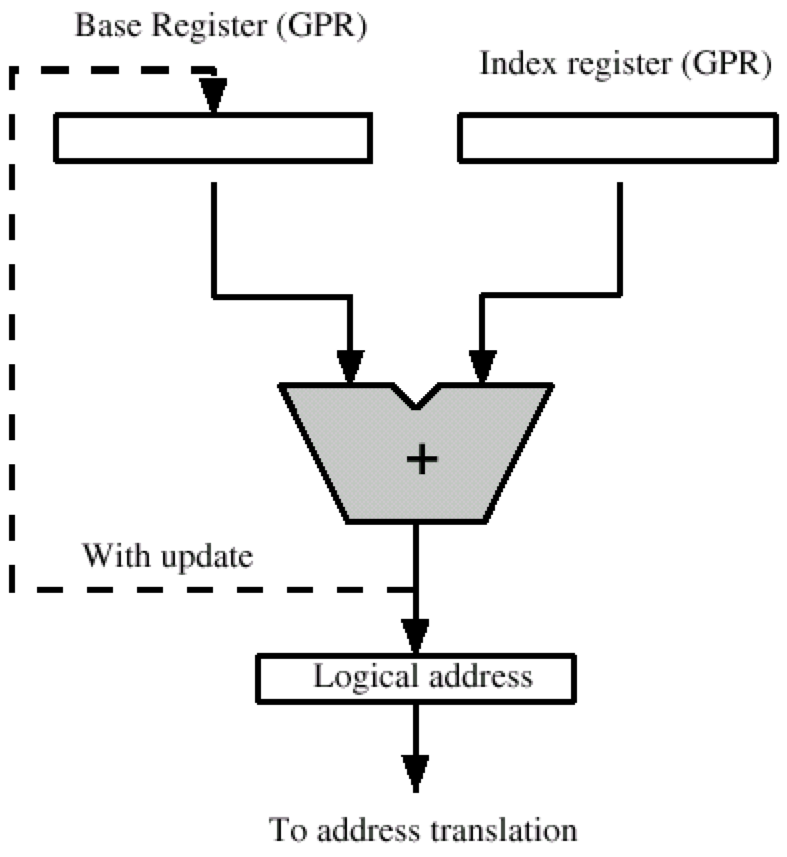
Mode	Algorithm
Load/Store Addressing	
Indirect	EA = (BR) + D
Indirect Indexed	EA = (BR) + (IR)
Branch Addressing	
Absolute	EA = I
Relative	EA = (PC) + I
Indirect	EA = (L/CR)
Fixed-Point Computation	
Register	EA = GPR
Immediate	Operand = I
Floating-Point Computation	
Register	EA = FPR

EA	=	effective address
(X)	=	contents of X
BR	=	base register
IR	=	index register
L/CR	=	link or count register
GPR	=	general-purpose register
FPR	=	floating-point register
D	=	displacement
I	=	immediate value
PC	=	program counter

PowerPC Memory Operand Addressing Modes



(a) Indirect Adressing



(b) Indirect Indexed Addressing

11.3 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
- Design issues
  - Instruction length
  - Allocation of bits
  - Fixed/Variable-length instructions

Instruction Length

- Affects and affected by :
  - Memory size
  - Memory organization
  - Bus structure
  - CPU complexity
  - CPU speed
- Trade off between powerful instruction repertoire and saving space
  - More opcodes and operands make shorter program possible
  - More addressing modes and address fields make larger address space possible
  - But longer instruction length may be wasteful

Allocation of Bits

- Design factors for allocation of bits
  - Number of addressing modes
    - If we need to specify it explicitly, some bits are needed
  - Number of operands
  - Register versus memory
    - Only a few bits are needed to specify the register
    - Most machines today have at least 32 general purpose registers
  - Number of register sets
    - General-purpose set + specialized set
  - Address range
    - Indirect addressing is favored to address larger space
  - Address granularity
    - Byte or word addressing
    - Byte addressing is convenient for character manipulation but requires more address bits

PDP-8

- One of the simplest instruction designs
  - 12-bit instructions on 12-bit words
    - 3-bit opcode and three type of instructions
    - Memory is divided into pages of 2^7 words each
  - Single GPR, accumulator
- Instruction format
  - Memory reference instructions : opcodes 0 through 5
    - Includes a page bit and indirect bit
  - I/O instructions : opcode 6
    - 6 bits used to select one of 64 devices
    - 3 bits to specify a particular command
  - Register reference instructions : opcode 7
    - Remaining bits are used to encode additional operations

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

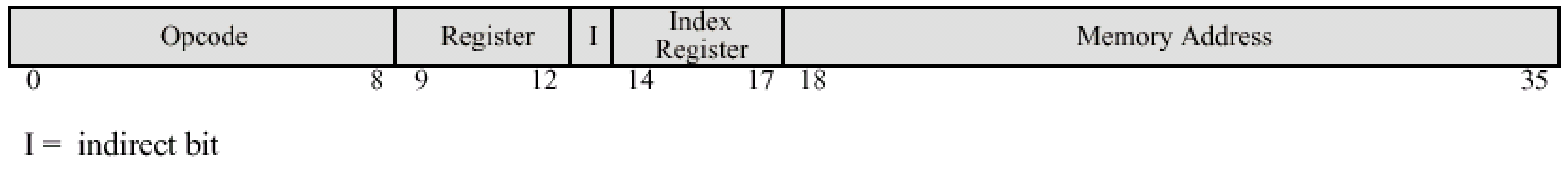
D/I	=	Direct/Indirect address	IAC	=	Increment ACcumulator
Z/C	=	Page 0 or Current page	SMA	=	Skip on Minus Accumulator
CLA	=	Clear Accumulator	SZA	=	Skip on Zero Accumulator
CLL	=	Clear Link	SNL	=	Skip on Nonzero Link
CMA	=	CoMplement Accumulator	RSS	=	Reverse Skip Sense
CML	=	CoMplement Link	OSR	=	Or 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

- Designed to be a large-scale time-shared system
- Design principles
  - Orthogonality
    - Other elements of an instruction are independent of the opcode
      - an address is always computed in the same way, independent of the opcode
  - Completeness
    - Each arithmetic data type should have a complete and identical set of operations
  - Direct addressing

PDP-10

- Instruction format
  - 36-bit word length and 36-bit instruction length

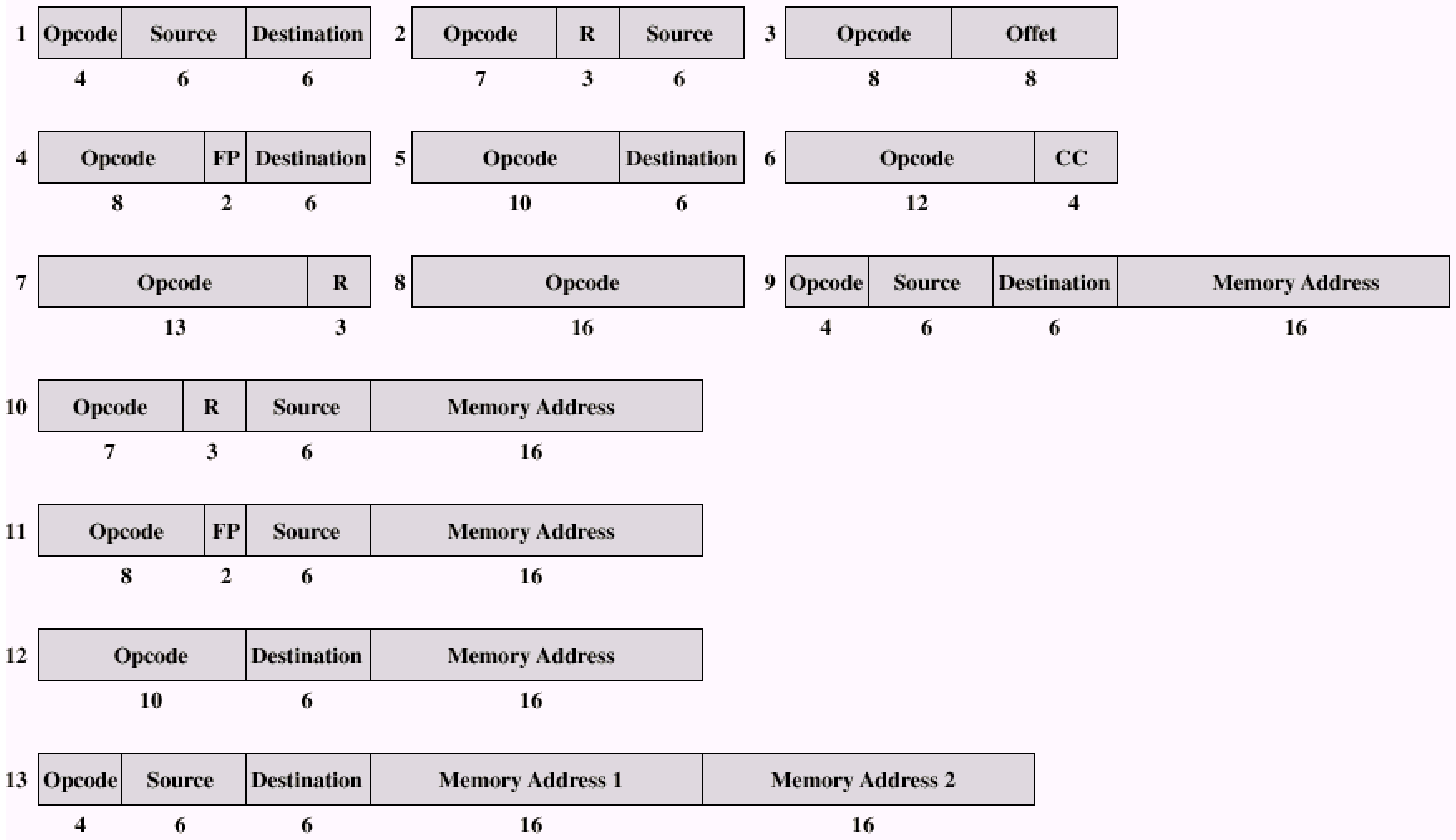


  - 9 bits for opcodes
    - 365 instructions are defined
  - Most instructions have two addresses
    - One is one of 16 GPRs
    - The other is 18-bit memory address

PDP-11

- 16-bit minicomputer
  - Has variable-length instructions
  - Has 16 GPRs, including PC and SP
- Instruction format
  - 13 different formats are used
    - Zero-, one-, and two-address instructions
  - Opcode lengths vary from 4 to 16 bits
  - 6 bits for register reference
    - 3 to identify the register and 3 to identify modes
  - Instructions are usually 16-bits long
    - For some instructions, one or two memory addresses are appended

PDP-11 Instruction Format


1 Opcode 4 | Source 6 | Destination 6
2 Opcode 7 | R 3 | Source 6
3 Opcode 8 | Offet 8
4 Opcode 8 | FP 2 | Destination 6
5 Opcode 10 | Destination 6
6 Opcode 12 | CC 4
7 Opcode 13 | R 3
8 Opcode 16
9 Opcode 4 | Source 6 | Destination 6 | Memory Address 16
10 Opcode 7 | R 3 | Source 6 | Memory Address 16
11 Opcode 8 | FP 2 | Source 6 | Memory Address 16
12 Opcode 10 | Destination 6 | Memory Address 16
13 Opcode 4 | Source 6 | Destination 6 | Memory Address 1 16 | Memory Address 2 16

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

- Design principles
  - All instructions should have the natural number of operands
  - All operands should have the same generality in specification
- Result is a highly variable instruction format
  - Instruction consists of a 1- or 2-byte opcode followed by from zero to six operand specifiers
    - Instructions from 1 to 37 bytes long
  - Instruction begins with a 1-byte opcode
    - FD and FF indicate an extended opcode
    - Second byte specify opcode

VAX

- Instruction with 6 operands
  - ADDP6 OP1, OP2, OP3, OP4, OP5, OP6
    - Instruction for adding two packed decimal numbers
    - OP1 and OP2 specify the length and starting address of one decimal string
    - OP3 and OP4 specify second string
    - Result is stored in a location specify by OP5 and OP6

11.4 Pentium Instruction Formats

- Instruction consists of
  - 0-4 optional prefixes
  - 1-2 byte opcode
  - Optional address specifier
    - Consists of ModR/m byte and Scale Index byte
  - Optional displacement
  - Optional immediate field

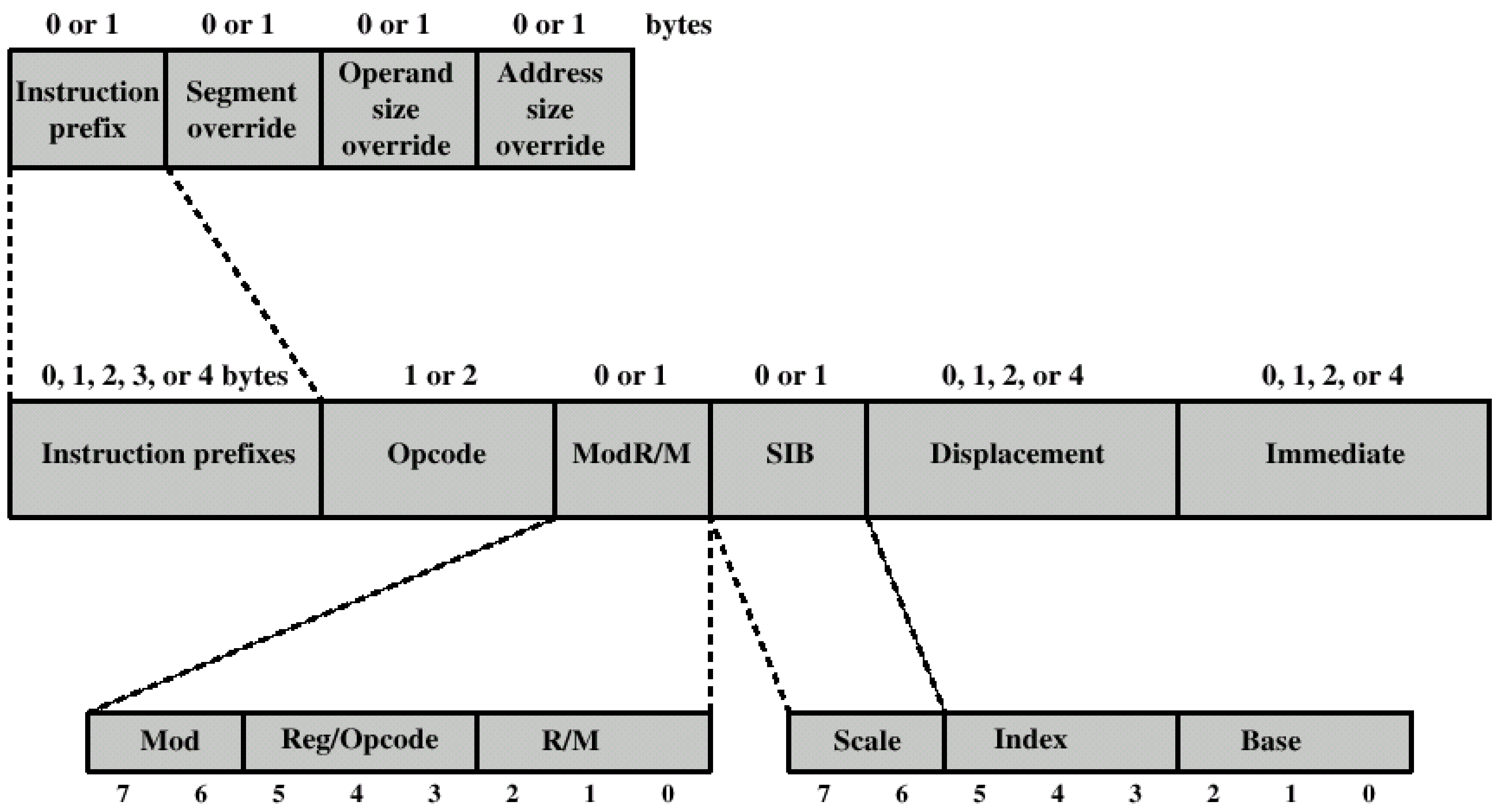
Pentium Instruction Formats

- Prefix bytes
  - Instruction prefixes
    - LOCK or one of repeat prefixes
    - LOCK is used to ensure exclusive use of shared memory in multiprocessor environments
    - REP, REPE, REPZ, REPNE, and REPNZ
      - Specify repeated operation on strings
      - Repeat until counter in CX goes zero or until the condition is met
  - Segment override
  - Address size
    - Switches between 32-bit and 16-bit address generation
  - Operand size
    - Switches between 32-bit and 16-bit operands

Pentium Instruction Formats

- Instruction
    - Opcode
    - ModR/m
        - Specify whether an operand is in a register or in memory
    - SIB
        - Specify fully the addressing mode
    - Displacement
        - When used, 8-, 16-, or 32-bit displacement field is added
    - Immediate
        - When used, 8-, 16-, or 32-bit operand is provided

Pentium Instruction Formats



PowerPC Instruction Formats

- Instruction formats
  - All instructions are 32 bits long and follow a regular format
  - First 6 bits specify the operation
    - May be extended elsewhere in the instruction for subcases
  - 5 bits for register references

PowerPC Instruction Formats

6 bits	5 bits	5 bits	16 bits		
Branch	Long Immediate			A	L
Br Conditional	Options	CR Bit	Branch Displacement	A	L
Br Conditional	Options	CR Bit	Indirect through Link or Count Register		L

(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

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