WEEK5

Chapter 11

Instruction Set Architecture
Addressing Modes and Formats

Types of Operand

- Addresses
- Numbers
  - Integer/floating point
- Characters
  - ASCII etc.
- Logical Data
  - Bits or flags

Memory Locations and Operations

- The (main) memory can be modeled as an array of millions of adjacent cells, each capable of storing a binary digit (bit), having value of 1 or 0.
- These cells are organized in the form of groups of fixed number, say n, of cells that can be dealt with as an atomic entity. An entity consisting of 8-bit is called a byte.
- The entity consisting of n-bit that can be stored and retrieved in and out of the memory using one basic memory operation is called a word.

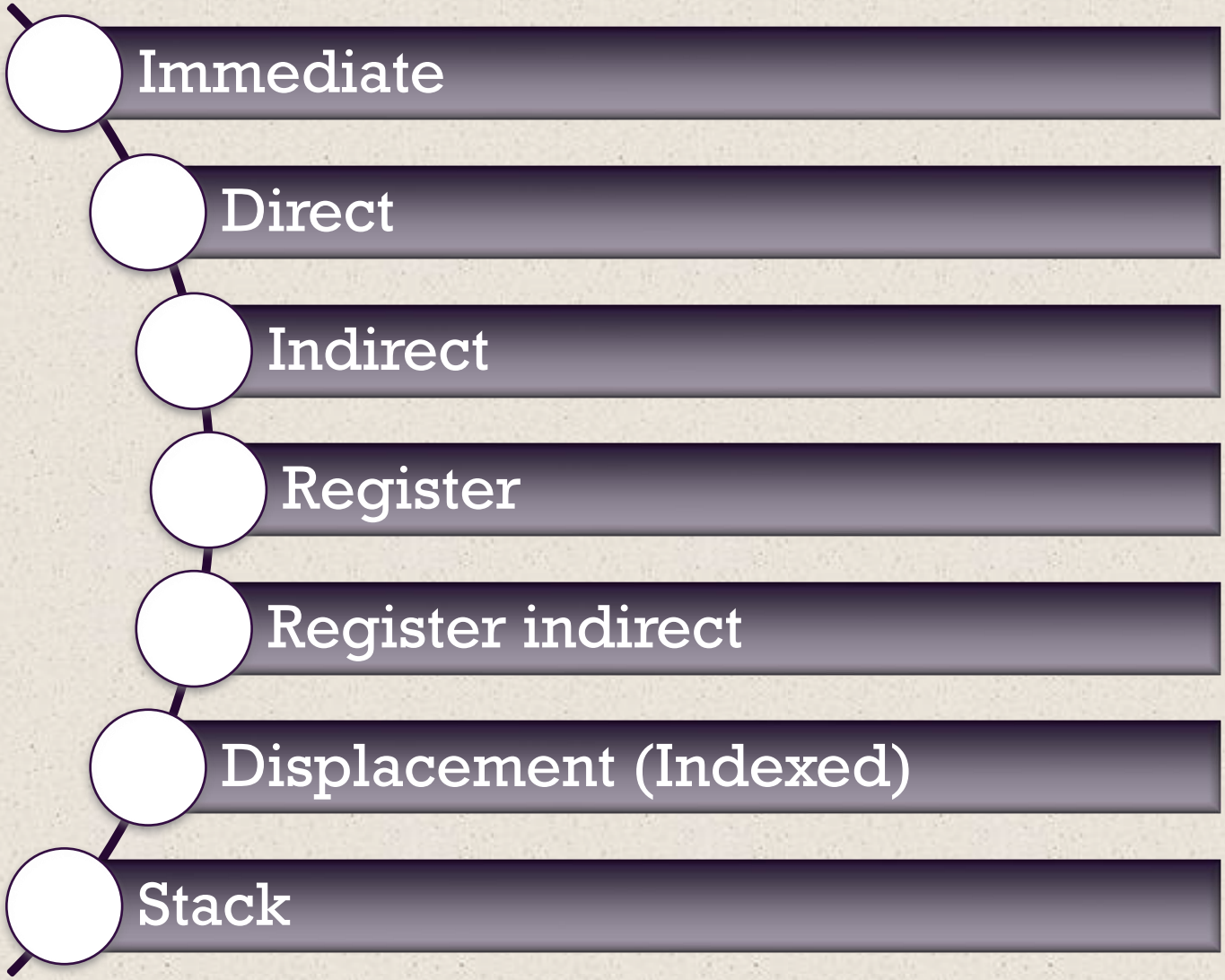
Memory Locations and Operations

- In order to be able to move a word in and out of the memory, a distinct address has to be assigned to each word.
- This address will be used to determine the location in the memory in which a given word is to be stored. This is called a memory ***write*** *operation.*
- Similarly, the address will be used to determine the memory location from which a word is to be retrieved from the memory. This is called a memory ***read*** operation.

Registers and Operations

- CPU must have some working space (temporary storage) Called registers
- Number and function vary between processor designs
- Top level of memory hierarchy
- User Visible Registers
  - Data register
  - Address register
- In order to be able to move a word in and out of the Register, a distinct address or register number has to be assigned.
- This address will be used to determine the specific register in which a given word is to be stored or read.

Addressing Modes



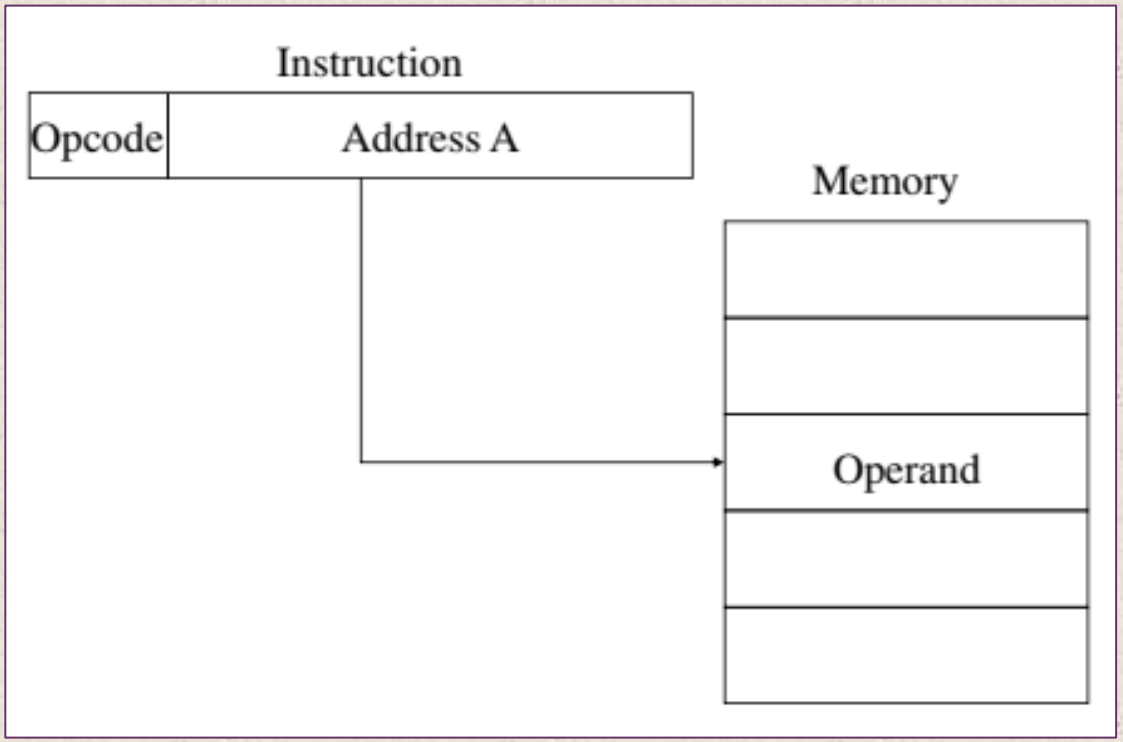
Immediate Addressing

- 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 and simple form of addressing
- Limited range
- The use of immediate addressing leads to poor programming practice. This is because a change in the value of an operand requires a change in every instruction that uses the immediate value of such operand.

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



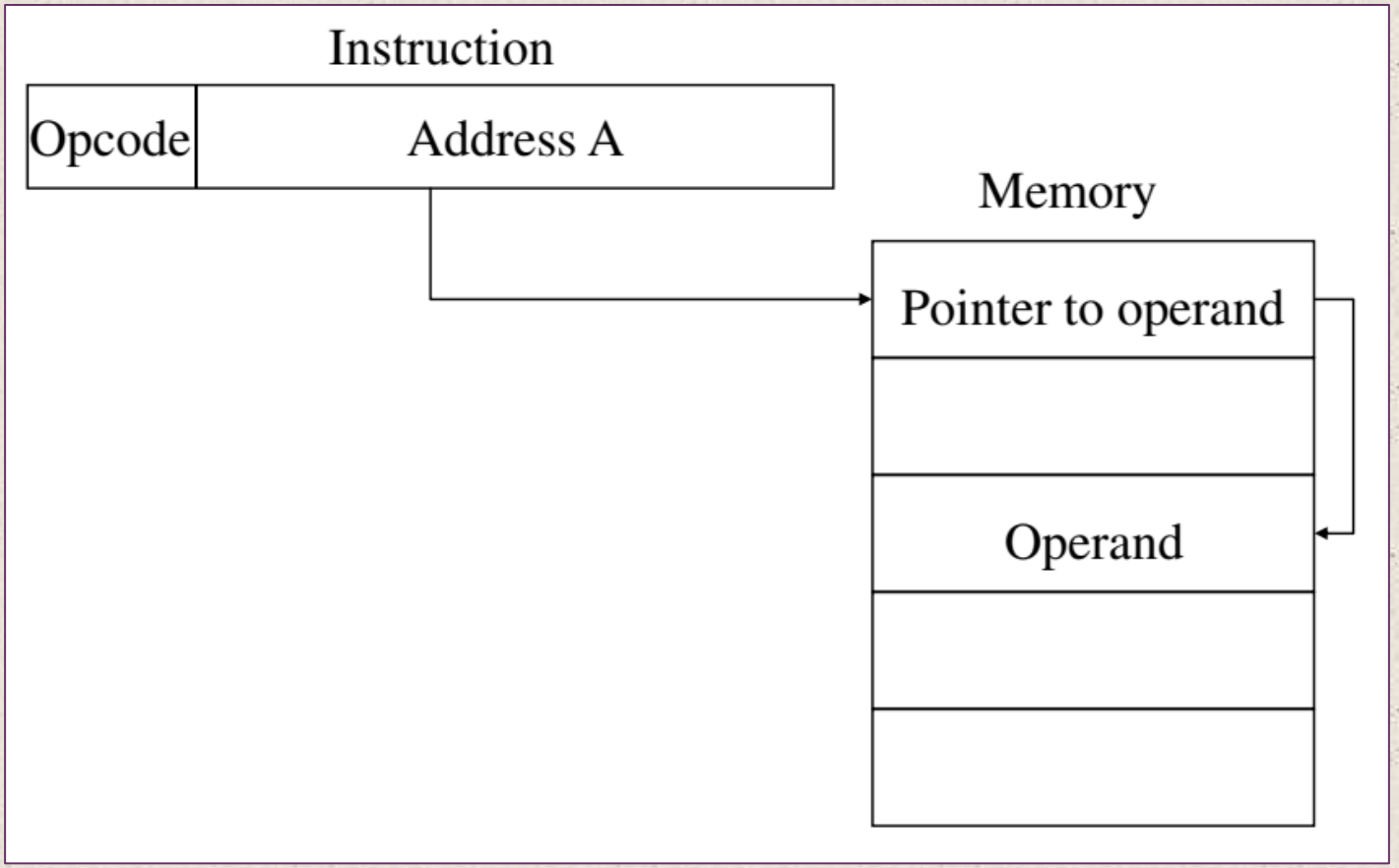
Indirect Addressing

- 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

- Large address space
- 2^n where n = word length
- May be nested, multilevel, cascaded
  - e.g. EA = (((A)))
- Instruction execution requires two memory references to fetch the operand
  - One to get its address and a second to get its value

Indirect Addressing Diagram



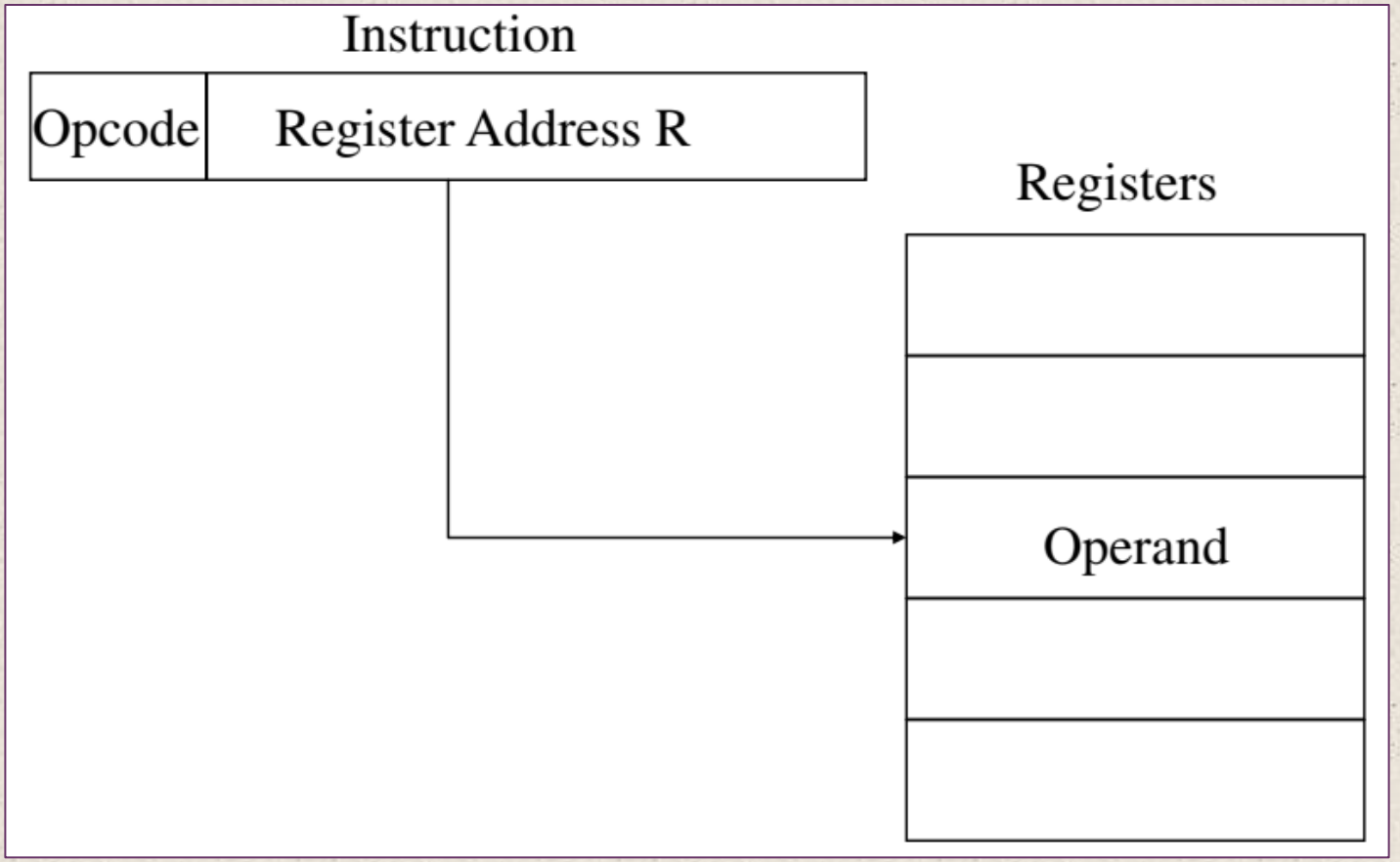
Register Addressing

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

Register Addressing

- 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

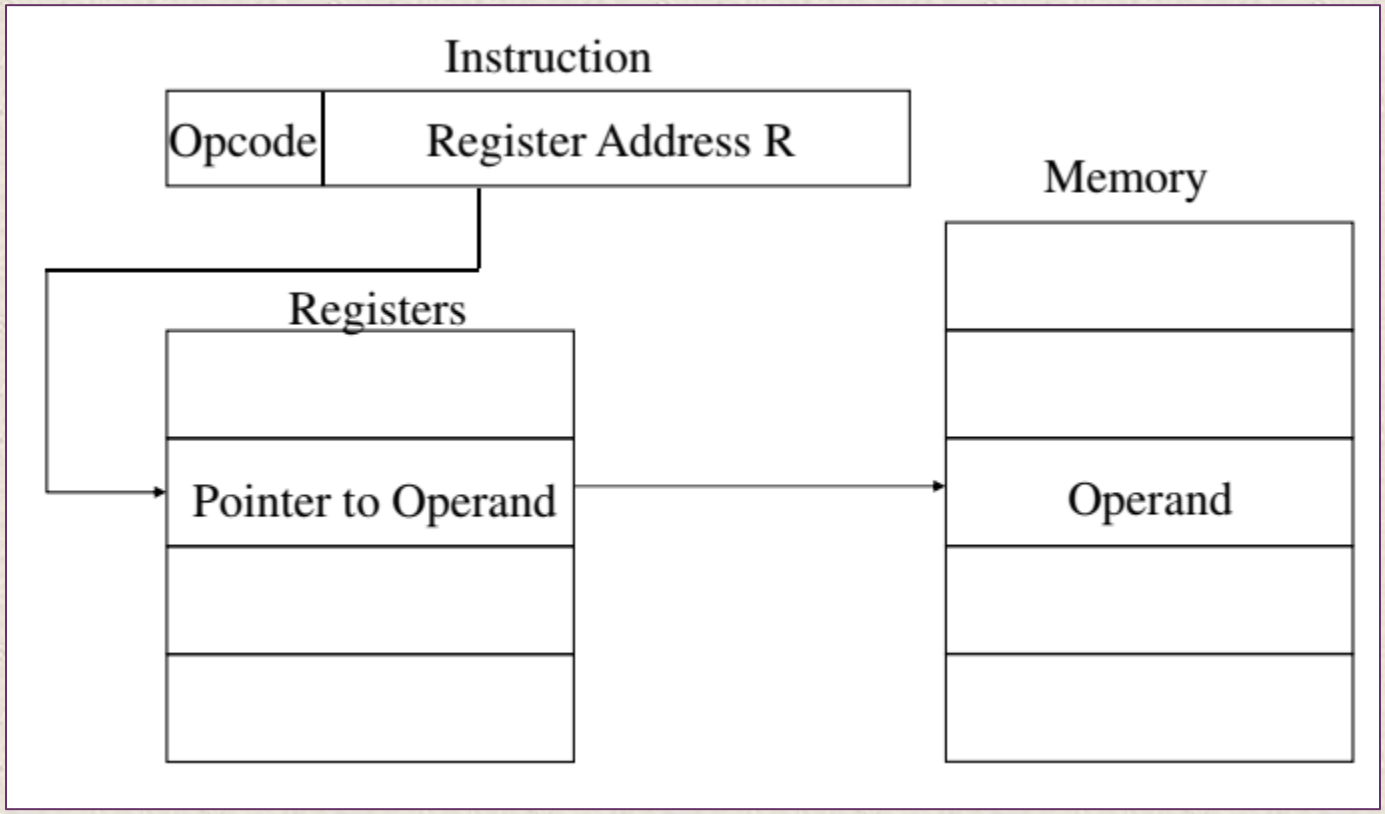
Register Addressing Diagram



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

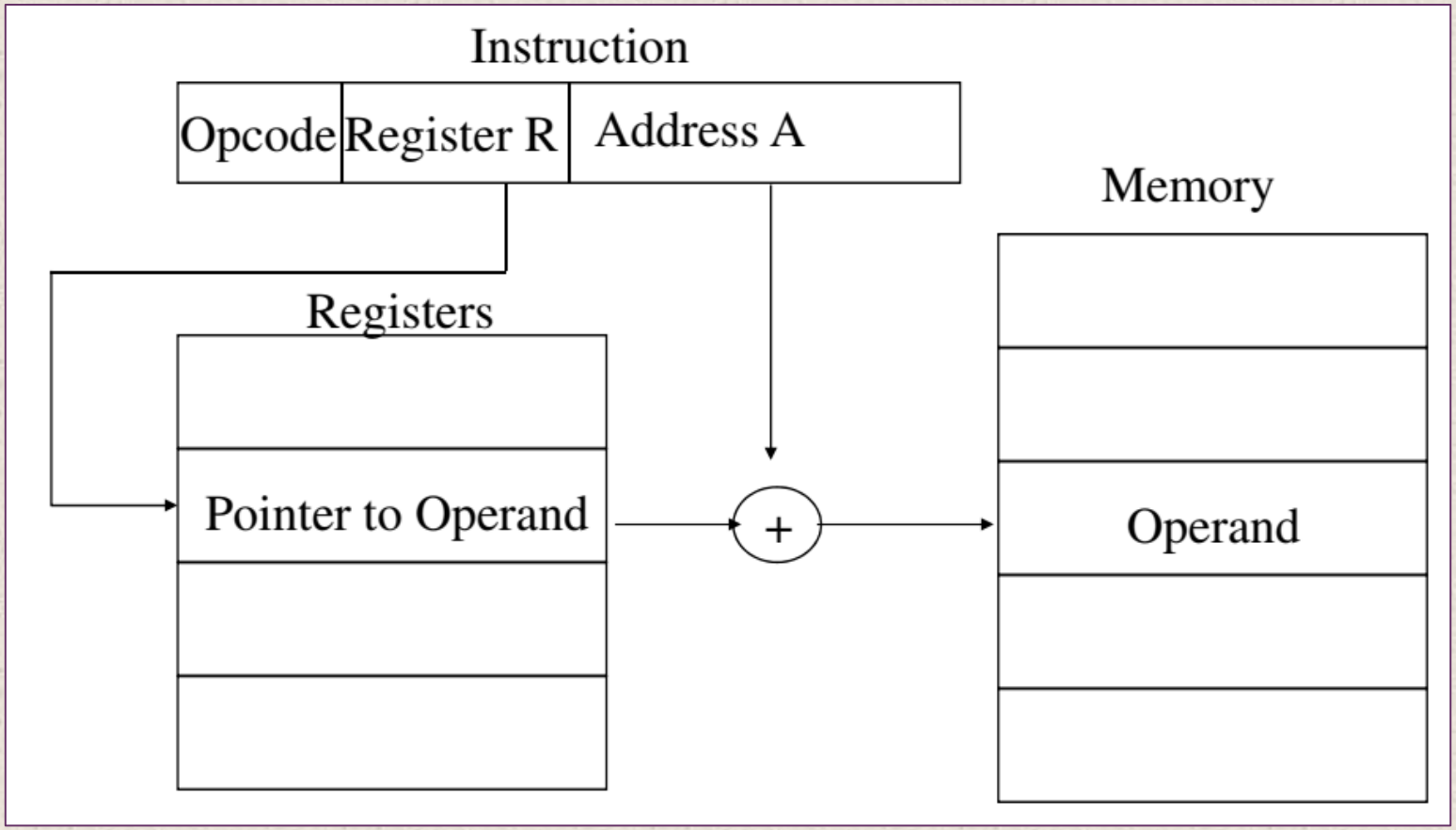
Register Indirect Addressing Diagram



Displacement Addressing

- Combines the capabilities of direct addressing and register indirect addressing
- EA = A + (R); address field hold two values
  - A = base value
  - R = register that holds displacement
  - or vice versa
- 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

Displacement Addressing Diagram



Relative Addressing

The implicitly referenced register is the program counter (PC)

$$EA = A + (PC)$$

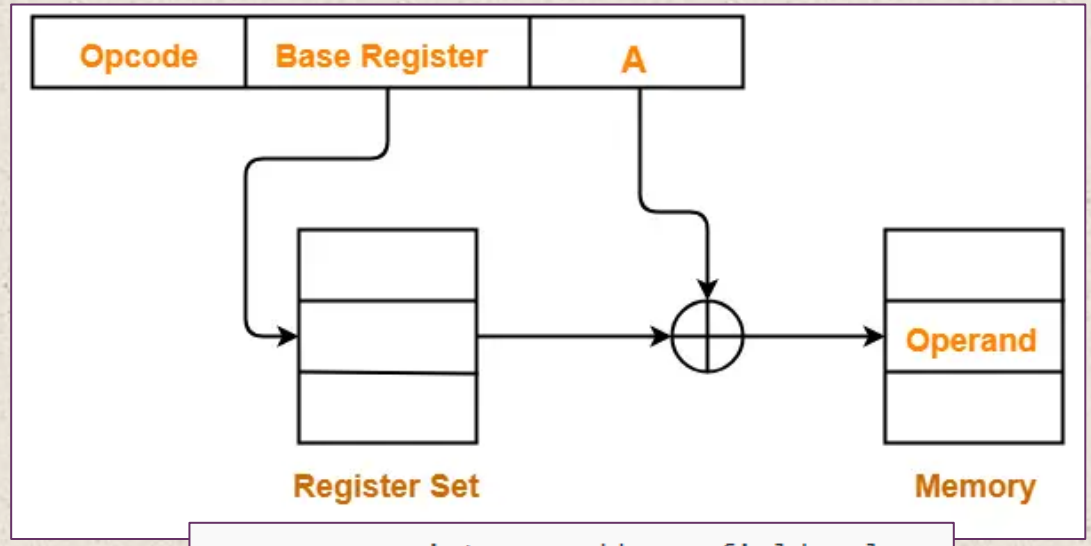
- The next instruction address is added to the address field to produce the EA
- Typically the address field is treated as a twos complement number for this operation
- Thus the effective address is a displacement relative to the address of the instruction

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

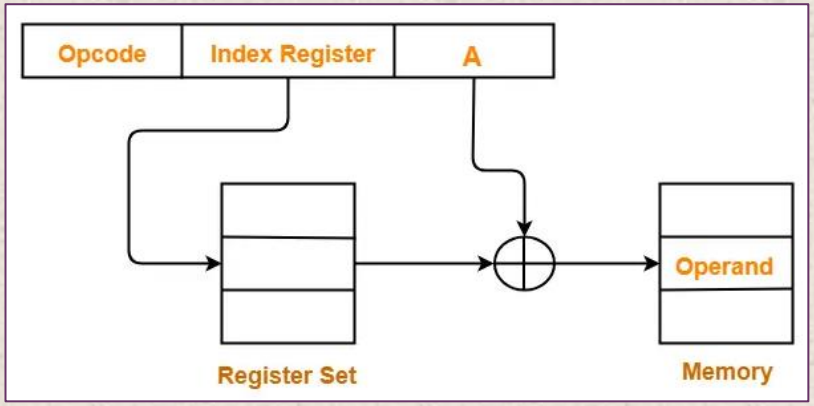
- Effective address of the operand is obtained by adding the content of base register with the address part of the instruction.
- EA= Content of Base Register + Address part of the instruction



EA= Base register + Address field value.
PC= Base register + Relative value.

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) ← (R) + 1
- Postindexing
  - Indexing is performed after the indirection
  - EA = (A) + (R)
- Preindexing
  - Indexing is performed before the indirection
  - EA = (A + (R))

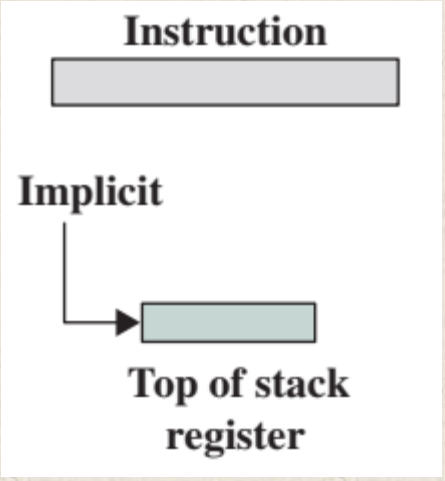


- **LD W0, [X1], 4**
  - LD W0, [X1]
  - ADD X1,X1,4

- **LD W0, [X1,4]**
  - ADD X1,X1,4
  - LD W0, [X1]

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
- ADD:
  - This instruction simply pops out symbols contained at the top of the stack.
  - The addition of those operands is performed.
  - The result so obtained after addition is pushed again at the top of the stack.



Example

13.1 Given the following memory values and a one-address machine with an accumulator, what values do the following instructions load into the accumulator?

- Word 20 contains 40.
- Word 30 contains 50.
- Word 40 contains 60.
- Word 50 contains 70.

a. LOAD IMMEDIATE 20
b. LOAD DIRECT 20
c. LOAD INDIRECT 20
d. LOAD IMMEDIATE 30
e. LOAD DIRECT 30
f. LOAD INDIRECT 30

a. 20
b. 40
c. 60
d. 30
e. 50
f. 70

Example

13.3 An address field in an instruction contains decimal value 14. Where is the corresponding operand located for

a. immediate addressing?
b. direct addressing?
c. indirect addressing?
d. register addressing?
e. register indirect addressing?

a) 14 (The address field).
b) Memory location 14.
c) The memory location whose address is in memory location 14.
d) Register 14.
e) The memory location whose address is in register 14.

Example

13.2 Let the address stored in the program counter be designated by the symbol X1. The instruction stored in X1 has an address part (operand reference) X2. The operand needed to execute the instruction is stored in the memory word with address X3. An index register contains the value X4. What is the relationship between these various quantities if the addressing mode of the instruction is (a) direct; (b) indirect; (c) PC relative; (d) indexed?

a) X3=X2
b) X3=(X2)
c) X3=X1+X2+1
d) X3=X2+X4

Example

Consider a 16-bit processor in which the following appears in main memory, starting at location 200:

200	Load to AC	Mode
201	500	
202	Next instruction	

The first part of the first word indicates that this instruction loads a value into an accumulator. The Mode field specifies an addressing mode and, if appropriate, indicates a source register; assume that when used, the source register is R1, which has a value of 400. There is also a base register that contains the value 100. The value of 500 in location 201 may be part of the address calculation. Assume that location 399 contains the value 999, location 400 contains the value 1000, and so on. Determine the effective address and the operand to be loaded for the following address modes:
a. Direct **b**. Immediate **c**. Indirect **d**. PC relative **e**. Displacement **f**. Register **g**. Register indirect **h**. Autoindexing with increment, using R1

a) 500 1100
b) 201 500
c) 1100 1700
d) 201+1+500 =702 1302
e) 500+100=600 1200
f) R1 400
g) 400 1000
h) 400 1000 >> R1 = 400+1

Addressing Modes	Applications
Immediate	• To initialize registers to a constant value
Direct and Register Direct	• To access static data • To implement variables
Indirect and Register Indirect	• To implement pointers because pointers are memory locations that store the address of another variable • To pass array as a parameter because array name is the base address and pointer is needed to point the address
Relative	• For program relocation at run time i.e. for position independent code • To change the normal sequence of execution of instructions • For branch type instructions since it directly updates the program counter
Index	• For array implementation or array addressing
Base Register	• For handling recursive procedures
Auto-increment And Auto-decrement	• For implementing loops • For stepping through arrays in a loop • For implementing a stack as push and pop

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