
Addressing Modes

Chapter 11

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
- 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

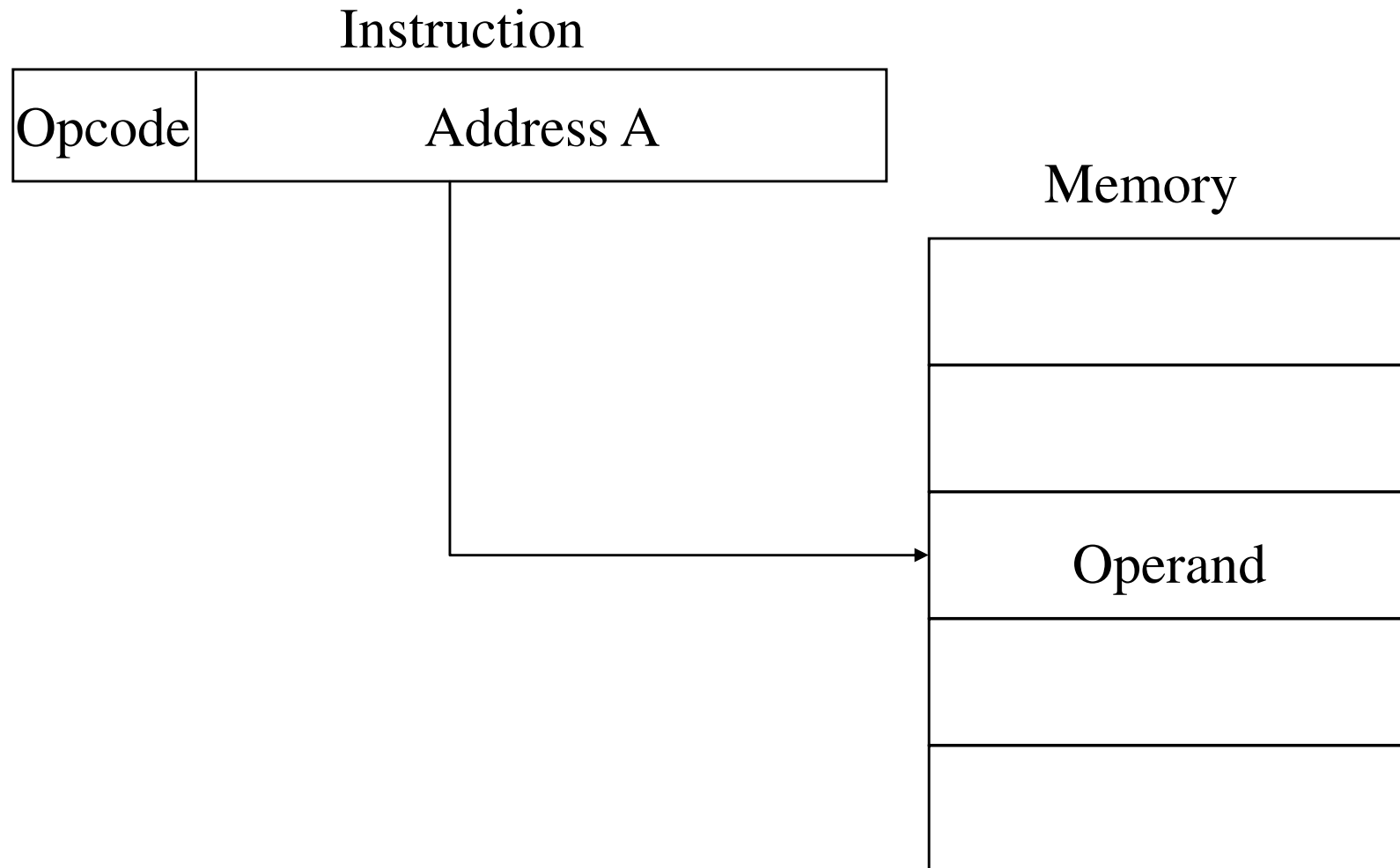


- No memory reference to fetch data
- Fast
- Limited range
- The use of immediate addressing leads to poor programming practice. This is because a change in the value of an operand requires a change in every instruction that uses the immediate value of such operand.

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



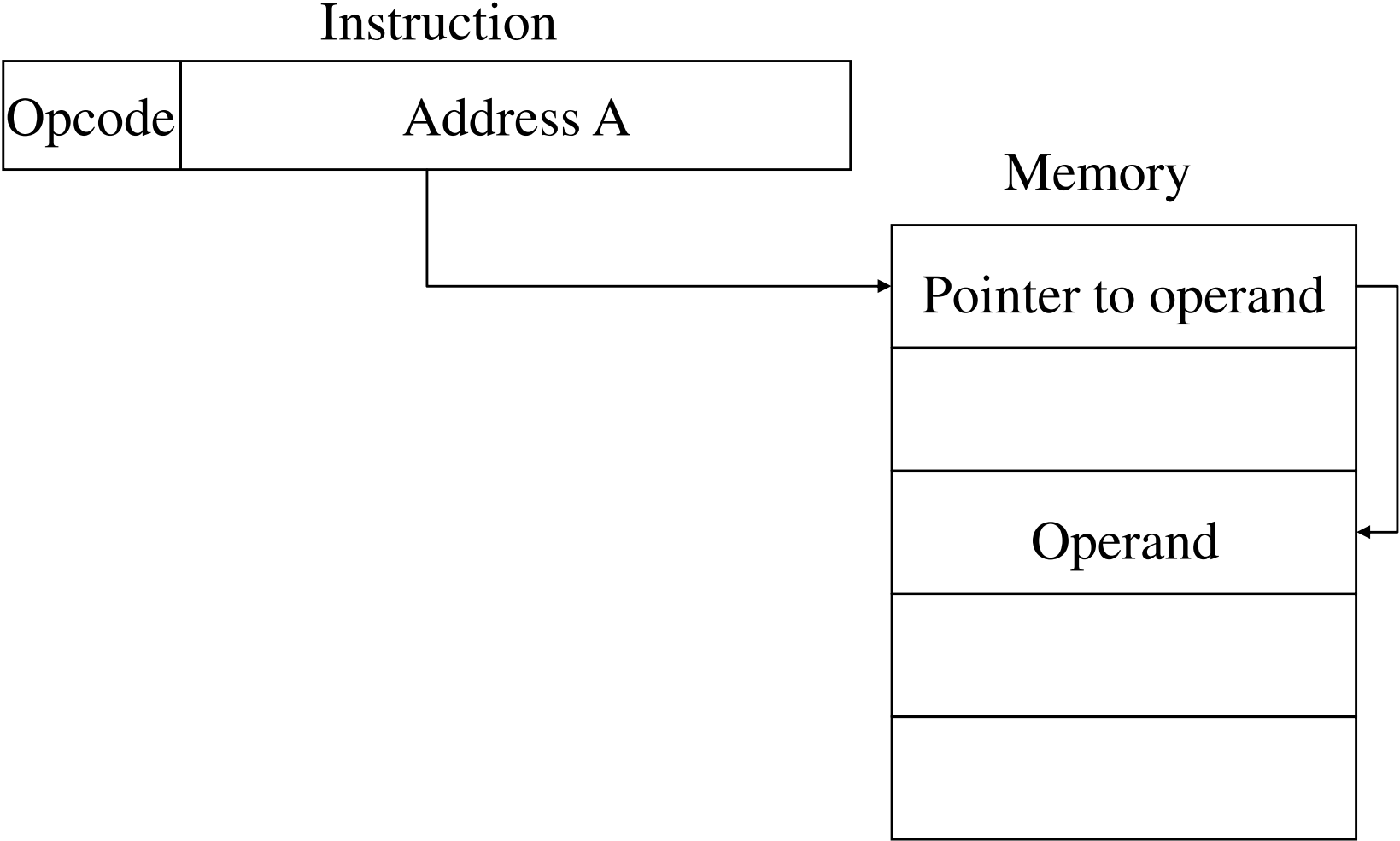
Indirect Addressing (1)

- 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



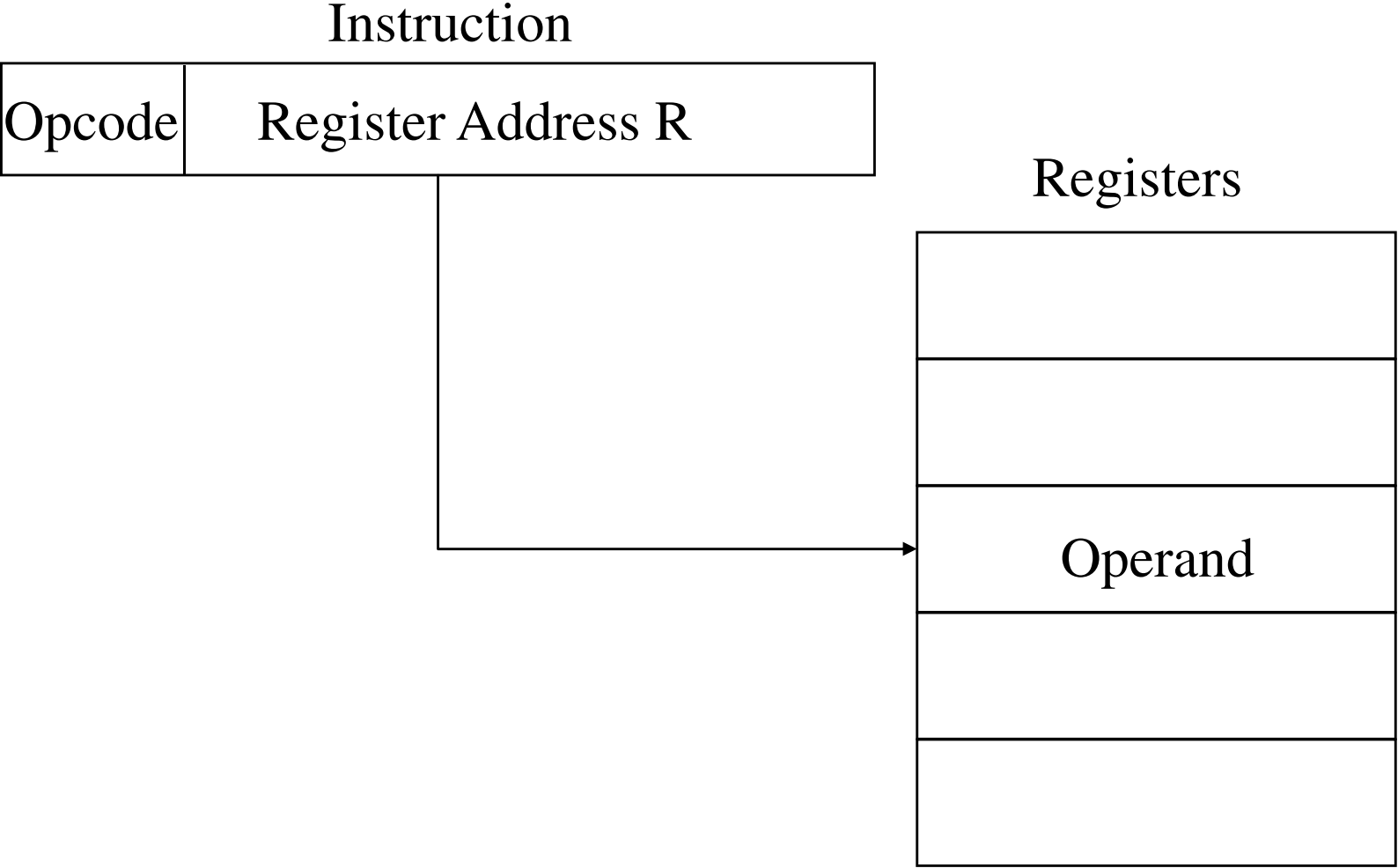
Register Addressing (1)

- Operand is held in register named in address field
- $EA = R$
- 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

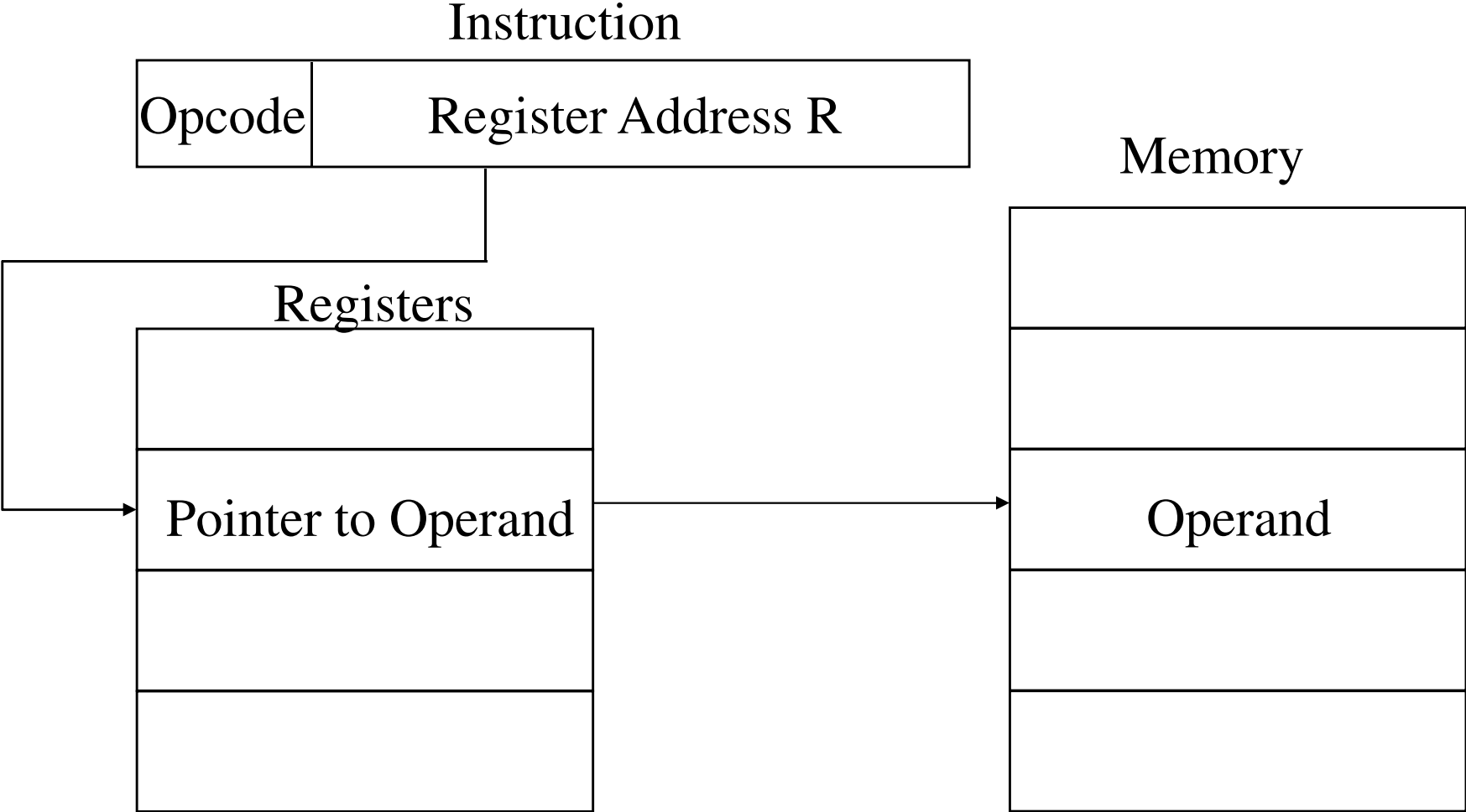
Register Addressing Diagram



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

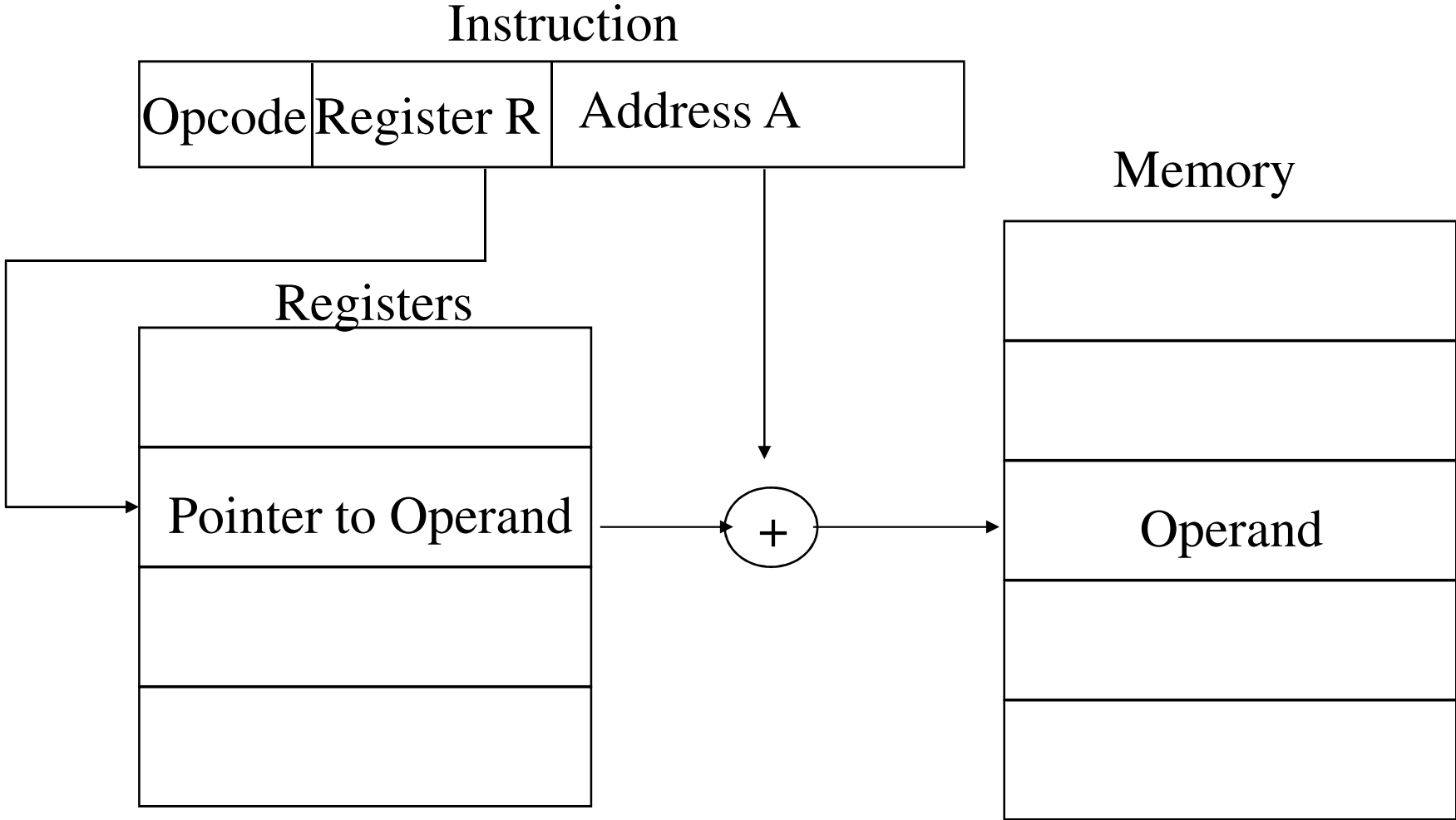
Register Indirect Addressing Diagram



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
- $R = \text{Program counter, PC}$
- $EA = A + (PC)$
- 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

- $A = \text{base}$
- $R = \text{displacement}$
- $EA = A + R$
- 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