Computer System Architecture

Course Code: CSE360

Lecture 10

Instruction Sets:

Addressing Modes and Formats

Key Points

- An operand reference in an instruction either contains the actual value of the operand (immediate) or a reference to the address of the operand
- A wide variety of addressing modes is used in various instruction sets
- These include direct (operand address is in address field), indirect (address field points to a location that contains the operand address), register, register indirect, and various forms of displacement, in which a register value is added to an address value to produce the operand address
- How is the address of an operand specified?
- The instruction format includes instruction length, fixed or variable length, number of bits assigned to opcode, and each operand reference, and how addressing mode is determined.

Addressing Modes

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

Immediate Addressing

- Operand value is present in the instruction
- Operand = A

  where, A is address field
- This mode can be used to define or set initial values of variables.
- e.g., ADD 5
  - Add 5 to contents of accumulator
  - 5 is operand
- No memory reference is required to fetch the data (operand), this saving memory or cache in the instruction cycle
- Fast
- Limited operand magnitude

Immediate Addressing Diagram

Instruction

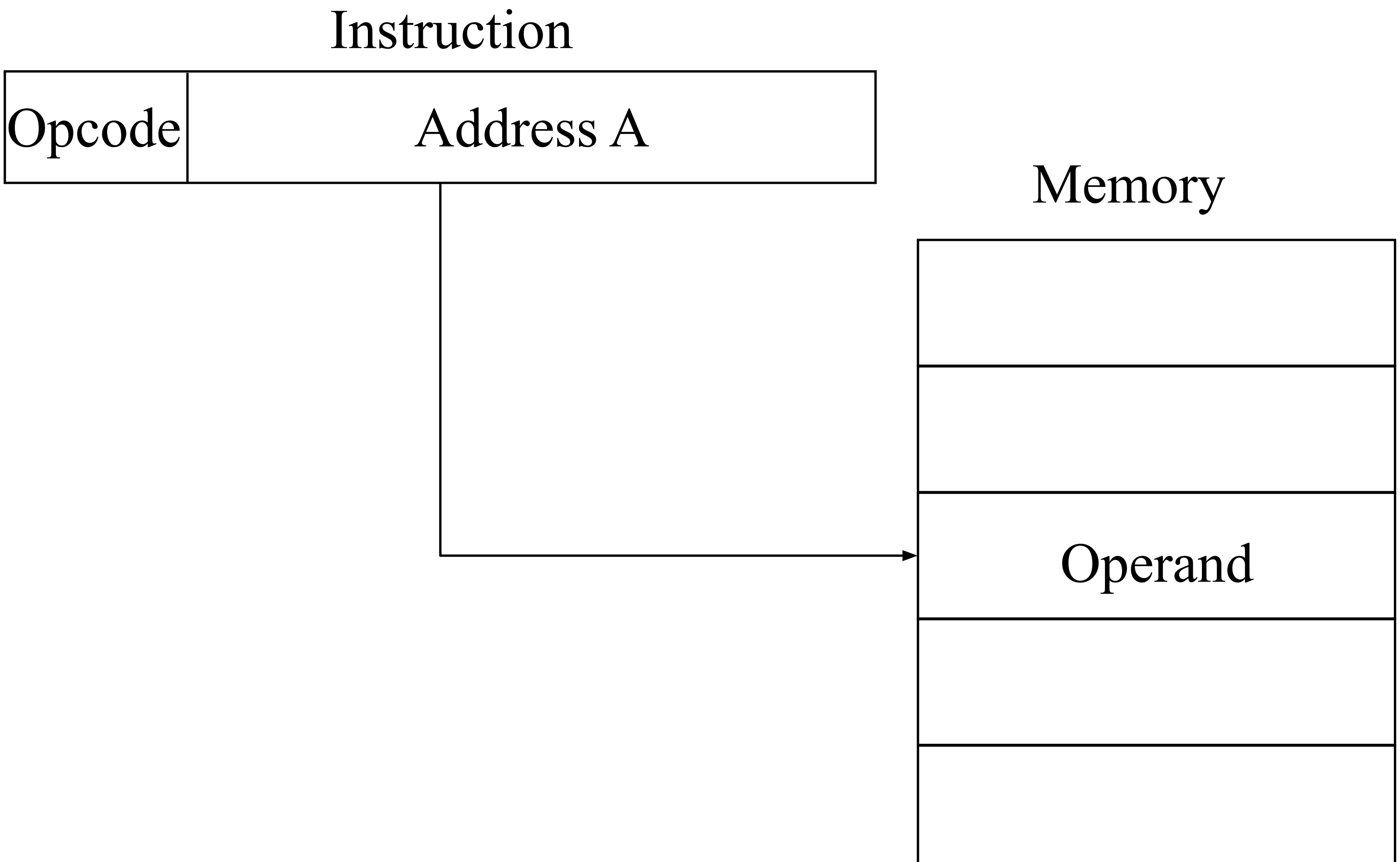
Opcode	Operand

Direct Addressing

- Address field contains address of operand in memory
- EA = A

  where EA is Effective Address
- e.g. ADD A
  - Add contents of 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



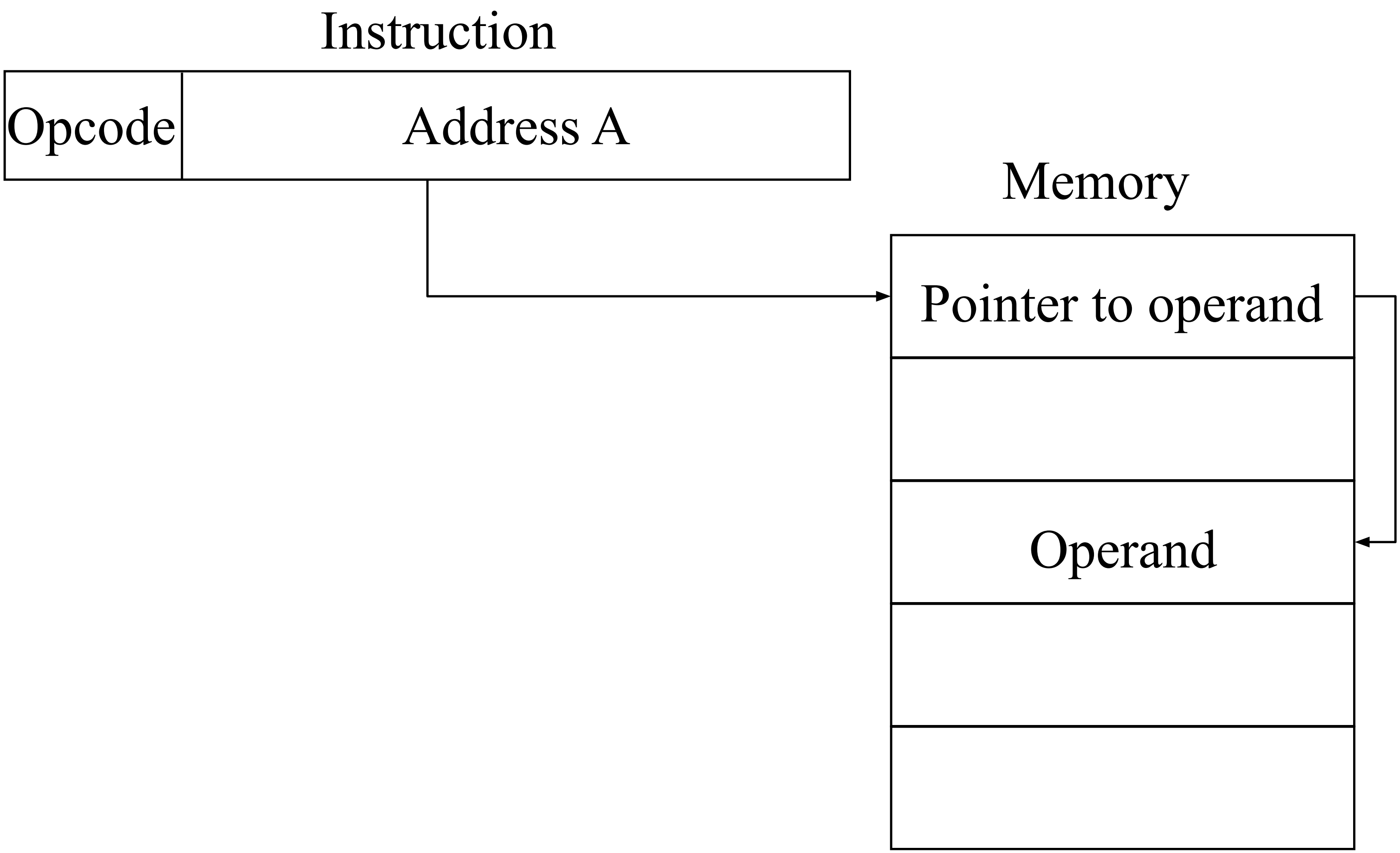
Indirect Addressing (1)

- With direct addressing, the length of the address field is usually less than the word length, thus limiting the address range
- One solution is to have the address field refer to (point to) the address of a word in memory, which in turn contains a full-length address of the operand.
- Simply, memory pointed to by address field contains the address of the operand
- EA = (A)
  - Look in A, find address (A), and look there for operand
  - Parentheses are to be interpreted as *contents of*
- e.g. ADD (A)
  - <u>Add contents of cell</u> pointed to by contents of A to accumulator

Indirect Addressing (2)

- Large address space is now available
- 2^n where n = word length
- May be nested, multilevel, cascaded
  - e.g. EA = (((A)))
- Multiple memory accesses could be required to find or fetch operand
- Hence, slower

Indirect Addressing Diagram



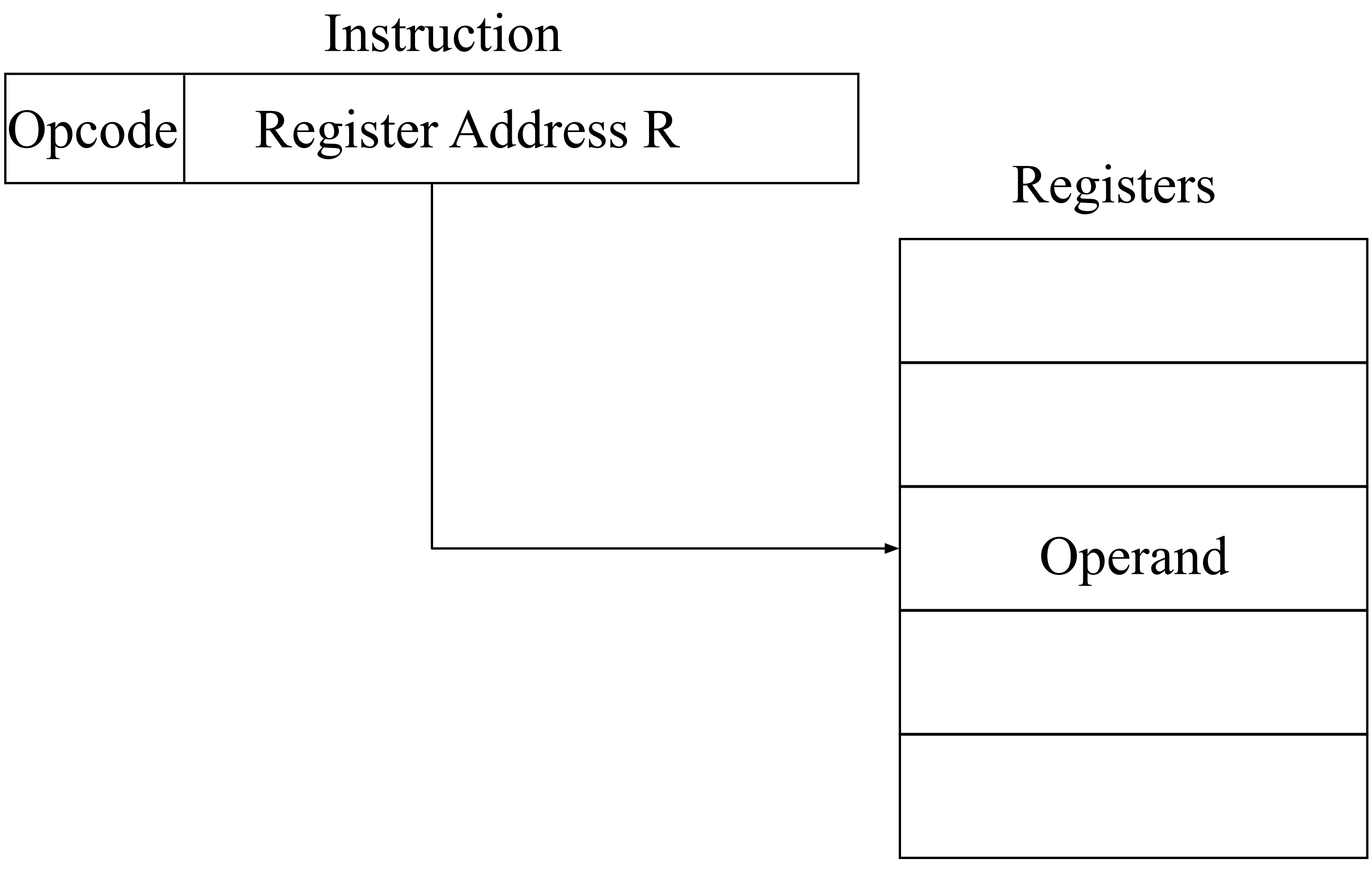
Register Addressing (1)

- Register addressing is similar to direct addressing; the only difference is that the address field refers to a register rather than a main memory address
- The address field refers to a register that contains the operand
- EA = R
- An address field that references registers will have from 3 to 5 bits, so total of 8 to 32 registers can be referenced
- Very small address field needed
  - Shorter instructions
  - Faster instruction fetch

Register Addressing (2)

- No memory access
- Very fast execution
- Very limited address space
- If every operand is brought into a register from main memory , operated on once, and then returned to main memory, then a wasteful approach is achieved
- If, instead, the operand in a register remains in use for multiple operations, then a real savings is achieved
- It is up to programmer to decide which values should remain in registers and which should be stored in main memory.
- Multiple registers helps performance
  - Requires good assembly programming or compiler writing
  - N.B. C programming
    - register int a;

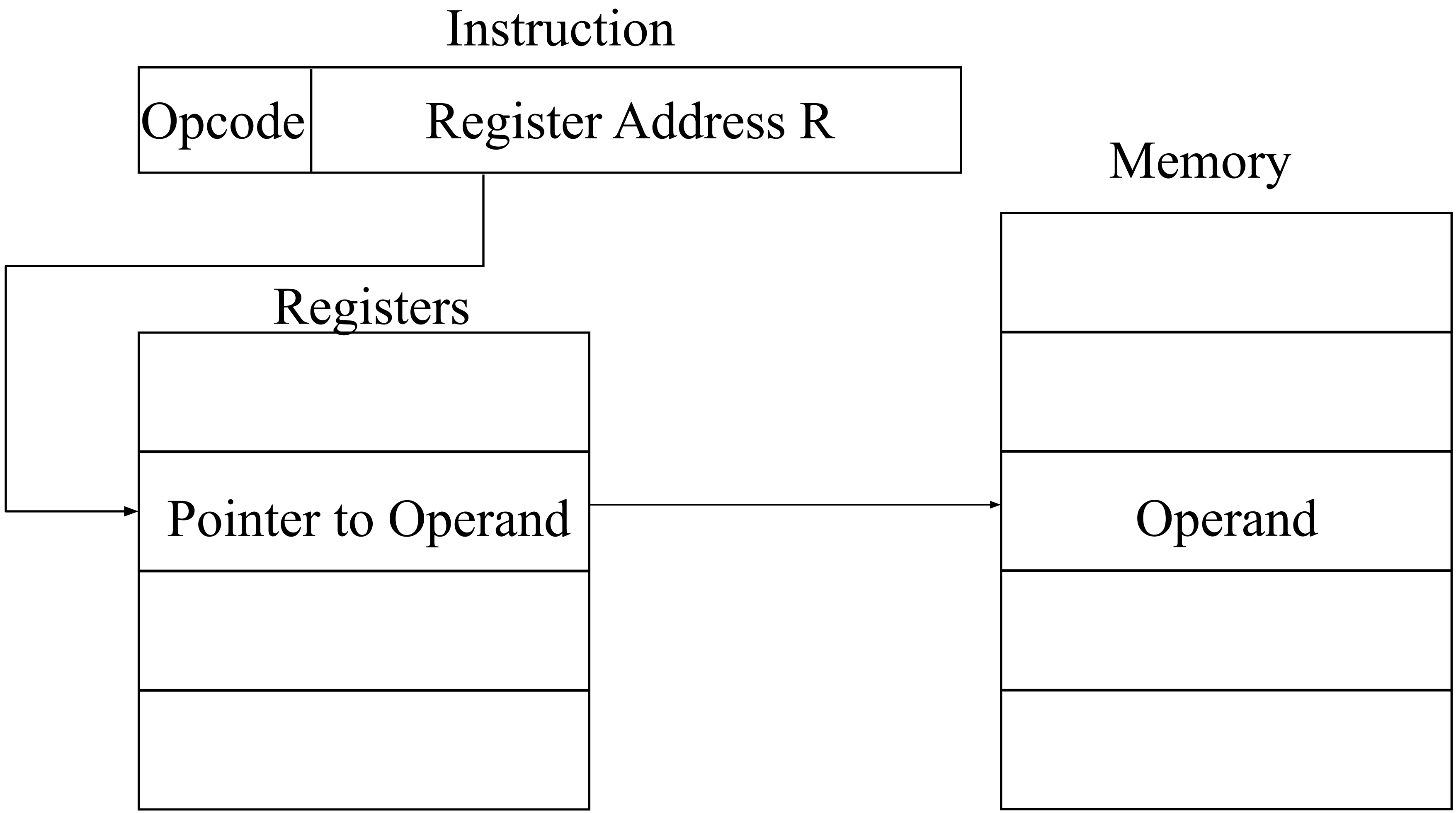
Register Addressing Diagram



Register Indirect Addressing

- Register indirect addressing is similar to indirect addressing
- The address field refers to a register R, which in turn points to the effective address of the operand in a memory
- EA = (R)
- Large address space (2^n)
- With direct addressing, the length of the address field is usually less than the word length, thus limiting the address range
- In register indirect addressing, address space limitation is overcome by having that <u>address field refer to the address of a word in a register, which in turn contains a full-length address of the operand in a memory.</u>
- In addition, register indirect addressing uses 1 less memory access than indirect addressing

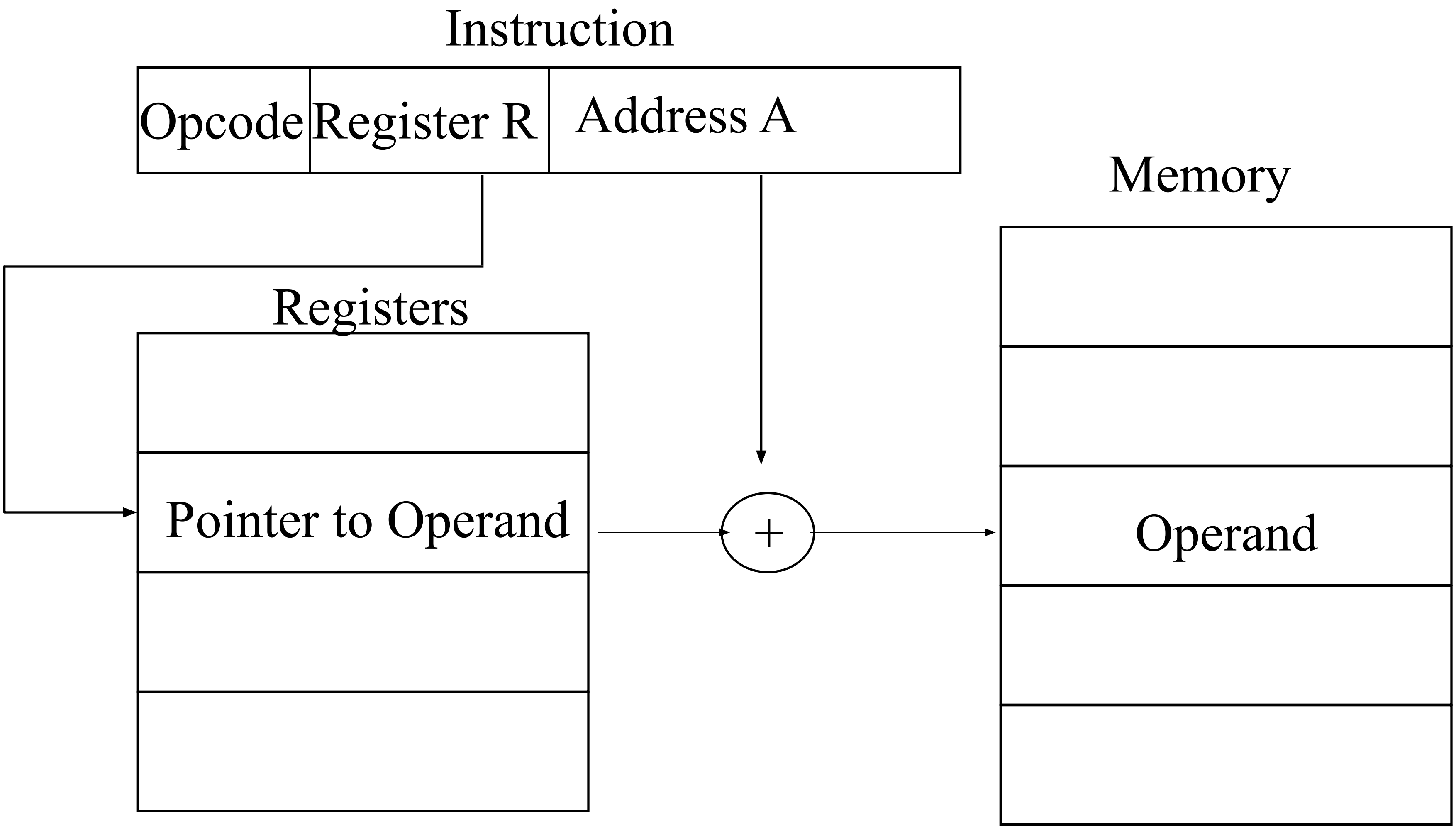
Register Indirect Addressing Diagram



Displacement Addressing

- The instruction has two address fields, at least one of which is explicit. The value contained in one address field (value = A) is used directly. <u>The other address field refers to a register whose contents are added to A to produce the effective address.</u>
- 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 address of operand A from current location pointed to by PC
- The implicitly referenced register is the program counter (PC). <u>That is, the next instruction address is added to the address field A to produce the EA.</u>

Base-Register Addressing

- The referenced register contains a memory address, and the address field contains a displacement
- A holds displacement
- R holds pointer to base address
- R may be explicit or implicit

Indexed Addressing

- There is a need to increment or decrement the index register after each reference to it. Some systems will automatically do this as part of the same instruction cycle, using autoindexing.
- A = base
- R = displacement
- EA = A + (R)
- Good for accessing arrays
  - EA = A + (R)
  - (R)=(R)+1

Combinations

- These are two forms of addressing, both of which involve indirect addressing and indexing. With **preindexing**, the indexing is performed before the indirection. With **postindexing**, the indexing is performed after the indirection.
- Postindex
  - $EA = (A) + (R)$
- Preindex
  - $EA = (A+(R))$

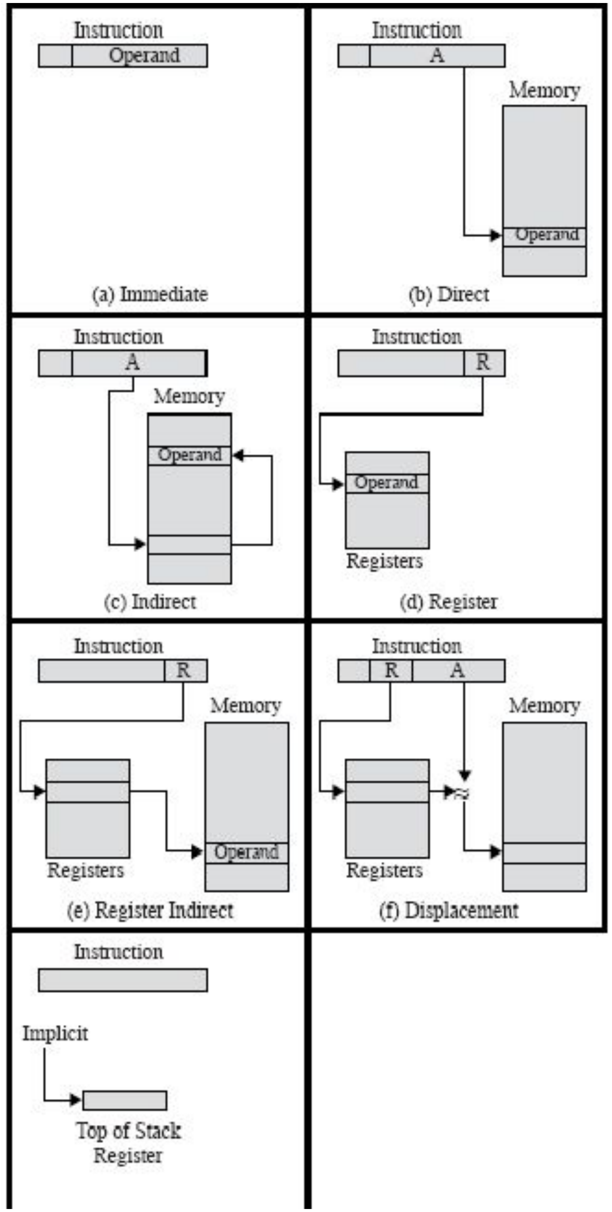
Stack Addressing

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

Basic 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

Basic Addressing Modes – contd….



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:** Sometimes an addressing mode can be indicated implicitly. In other cases, the addressing modes must be explicit, and one or more mode bits will be needed.
- **Number of operands:** Typical instructions on today's machines provide for two operands. Each operand address in the instruction might require its own mode indicator, or the use of a mode indicator could be limited to just one of the address fields.
- **Register versus memory:** The more that registers can be used for operand references, the fewer bits are needed.
- **Number of register sets:** One advantage of using multiple register sets is that, for a fixed number of registers, it requires fewer bits in the instruction.
- **Address range:** <u>For addresses that reference memory, the range of addresses that can be referenced is related to the number of address bits.</u> Because this imposes a severe limitation, direct addressing is rarely used. With displacement addressing, the range is opened up to the length of the address register.
- **Address granularity:** <u>In a system with 16- or 32-bit words, an address can reference a word or a byte at the designer's choice.</u> Byte addressing is convenient for character manipulation but requires, for a fixed-size memory, more address bits.

Advantages and disadvantages of variable length instruction format

Advantages:

- It easy to provide a large range of opcodes, with different opcode lengths.
- Addressing can be more flexible, with various combinations of register and memory references plus addressing modes.

Disadvantages:

An increase in the complexity of the CPU.