Addressing Modes

Muhammad Afzaal
m.afzaal@nu.edu.pk

Book Chapter

- “Computer Organization and Architecture”
- Author “William Stallings”
- 8th Edition
- Chapter 11
  - Section 11.1

Addressing Modes

- The set of mechanism by which an instruction can specify how to obtain its operands
- An operand address provides the location where the data to be processed is stored
- In an assembly instruction, the destination may be a register or memory location
- Source may be an immediate value or the address of the source data

Addressing Modes

- Immediate
- Direct
- Indirect
- Register
- Register Indirect
- Displacement
- Stack

Useful Notations

- In the next slides, we use the following notations
  - **A** denotes the contents of an address field in the instruction
  - **R** denotes the contents of an address field in the instruction that refers to a register
  - **EA** is the actual effective address of the location containing the referenced operand
  - **[X]** denotes the contents of memory location X or register X

Immediate Addressing (1/2)

- Simplest form of addressing where operand value is present in the instruction
- No memory reference other than instruction fetch is required to obtain the operand

```
Operand = A
```

Immediate Addressing (2/2)

- Example

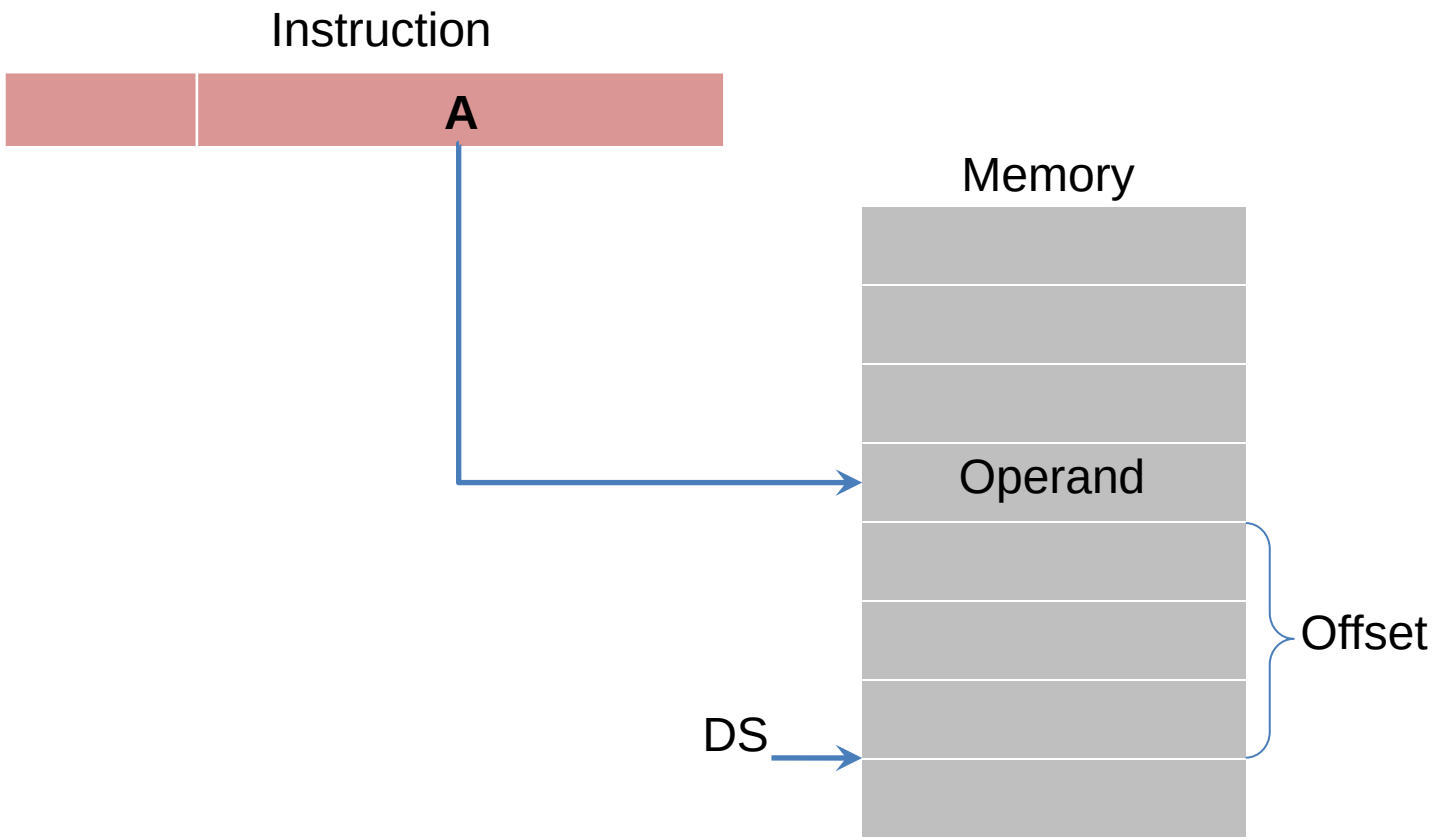
```
.code
   MOV AL, 10h
```

Direct Addressing (1/3)

- Direct access to the data segment is required in direct addressing mode
- Address field contains the effective address of operand
- Base address of segment and offset of storage location is required

$$EA = A$$

Direct Addressing (2/3)



Direct Addressing (3/3)

- Example

```
.data
    val DB 10h
.code
    MOV al, val
    MOV al, [102h] ;same as above
```

Indirect Addressing (1/3)

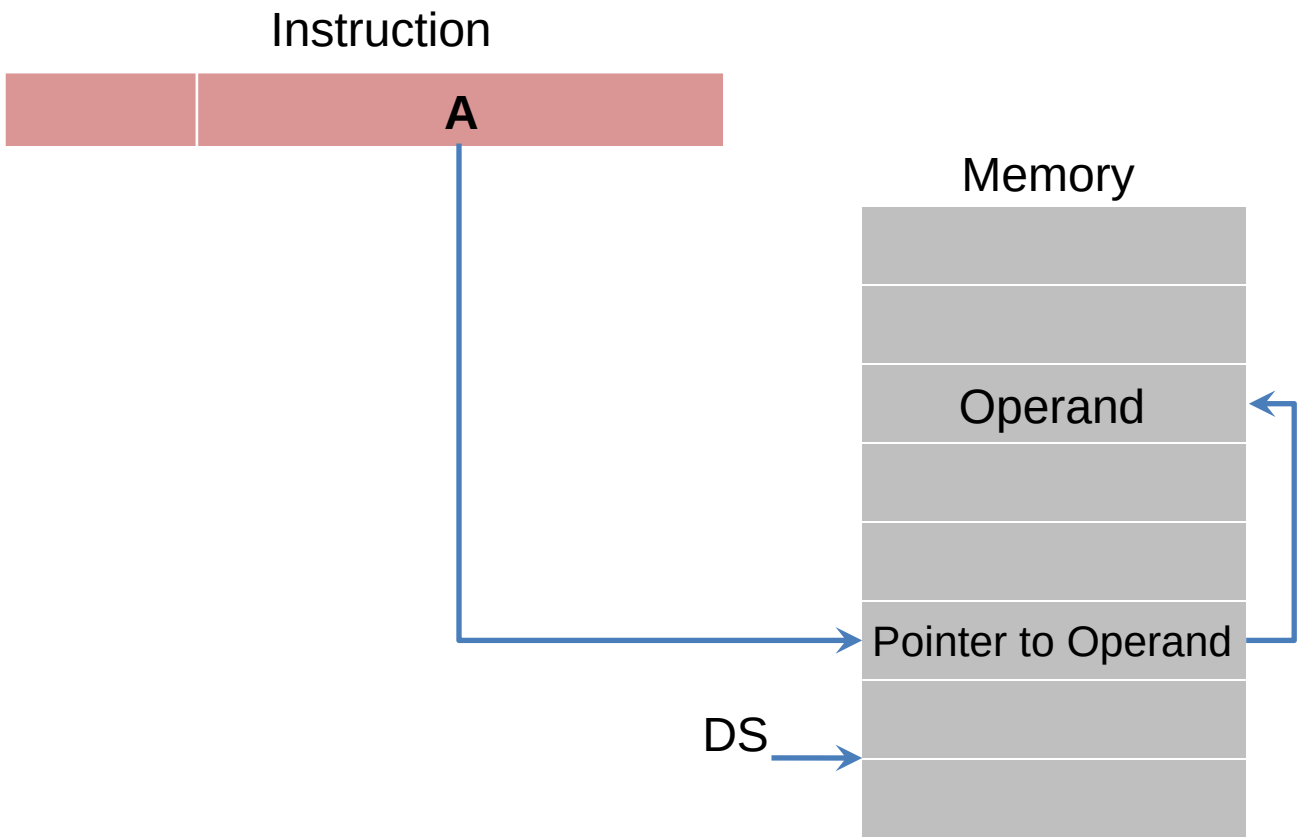
- Memory cell pointed to by address field contains the address of the operand
- One major advantage is to reference more words if address field is less than word length

$$EA = [A]$$

- Indirect addressing can be nested or cascaded

$$EA = [[A]]$$

Indirect Addressing (2/3)



Indirect Addressing (3/3)

- Example

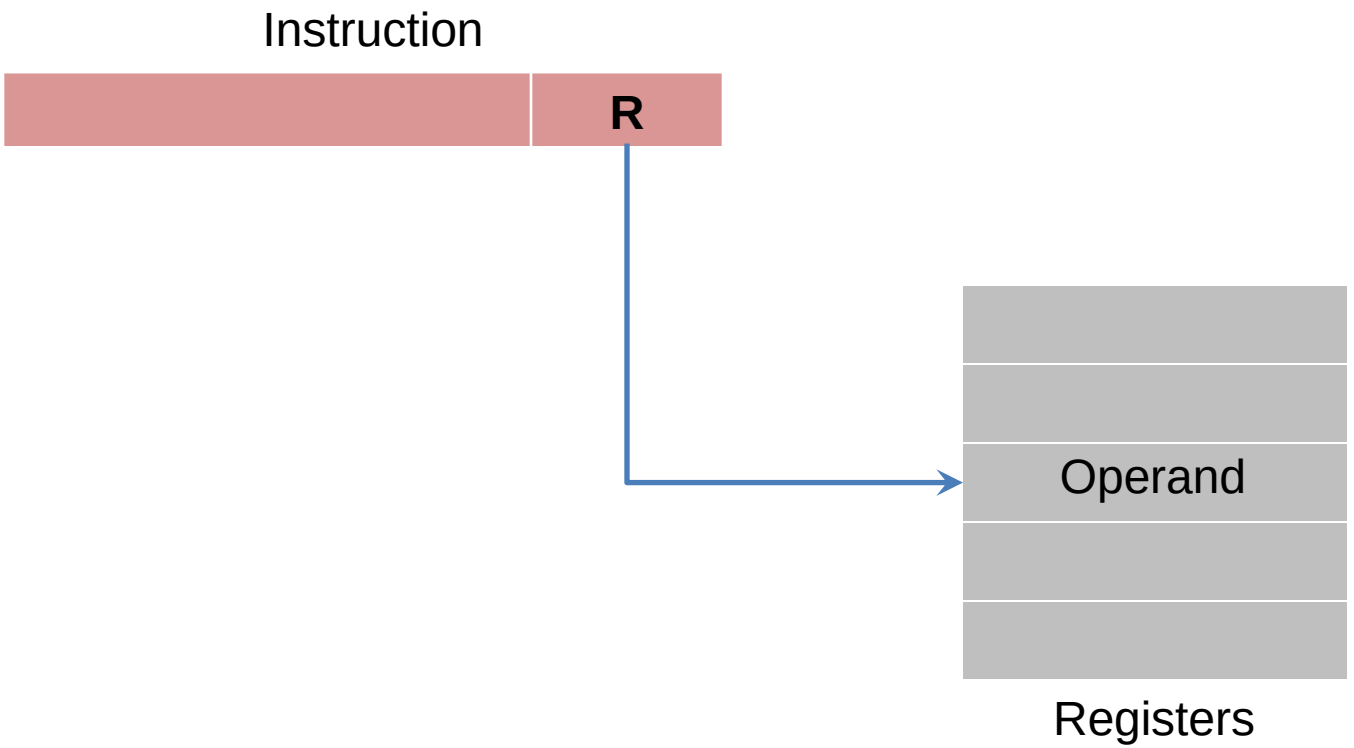
```
.data
    val DB 10h
    vptr DW val
.code
    MOV si, vptr
    MOV al, [si]
```

Register Addressing (1/3)

- Similar to direct addressing
- In this mode, a register contains the operand
- Depending upon the instruction, the register may be the first operand, the second operand or both
- Processing data between register does not involve memory, so it provides fastest processing of data

$$EA = R$$

Register Addressing (2/3)



Register Addressing (3/3)

- Example

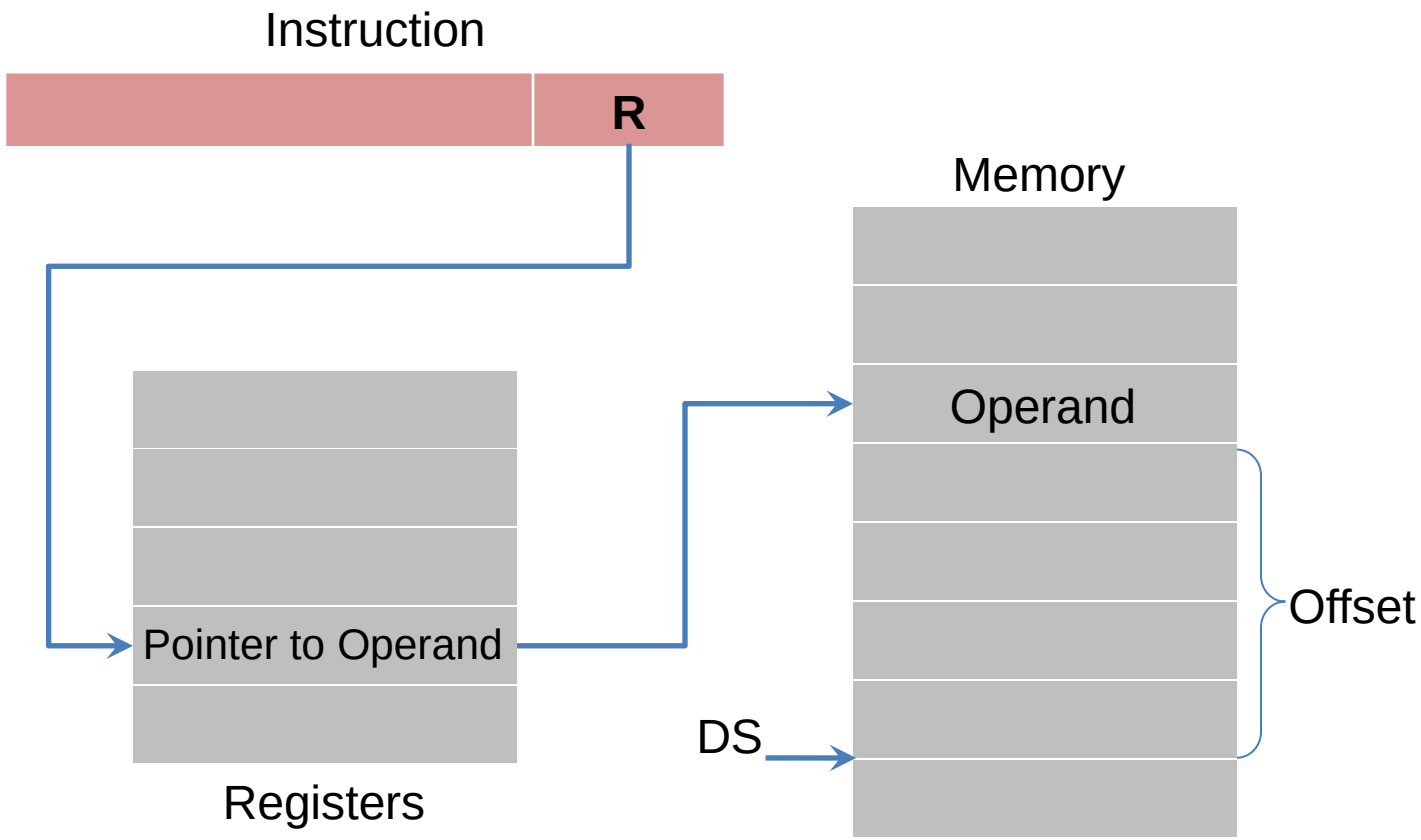
```
.data
    val DB 10h
.code
    MOV al, ah
```

Register Indirect Addressing (1/3)

- Similar to direct addressing that offset is combined with base address of `DS` to get physical address
- Operand is in memory cell pointed to by contents of register R
- Offset lies in a pointer register (`BX` or `BP`) or index register (`SI` or `DI`)

```
EA = [R]
```

Register Indirect Addressing (2/3)



Register Indirect Addressing (3/3)

- Example

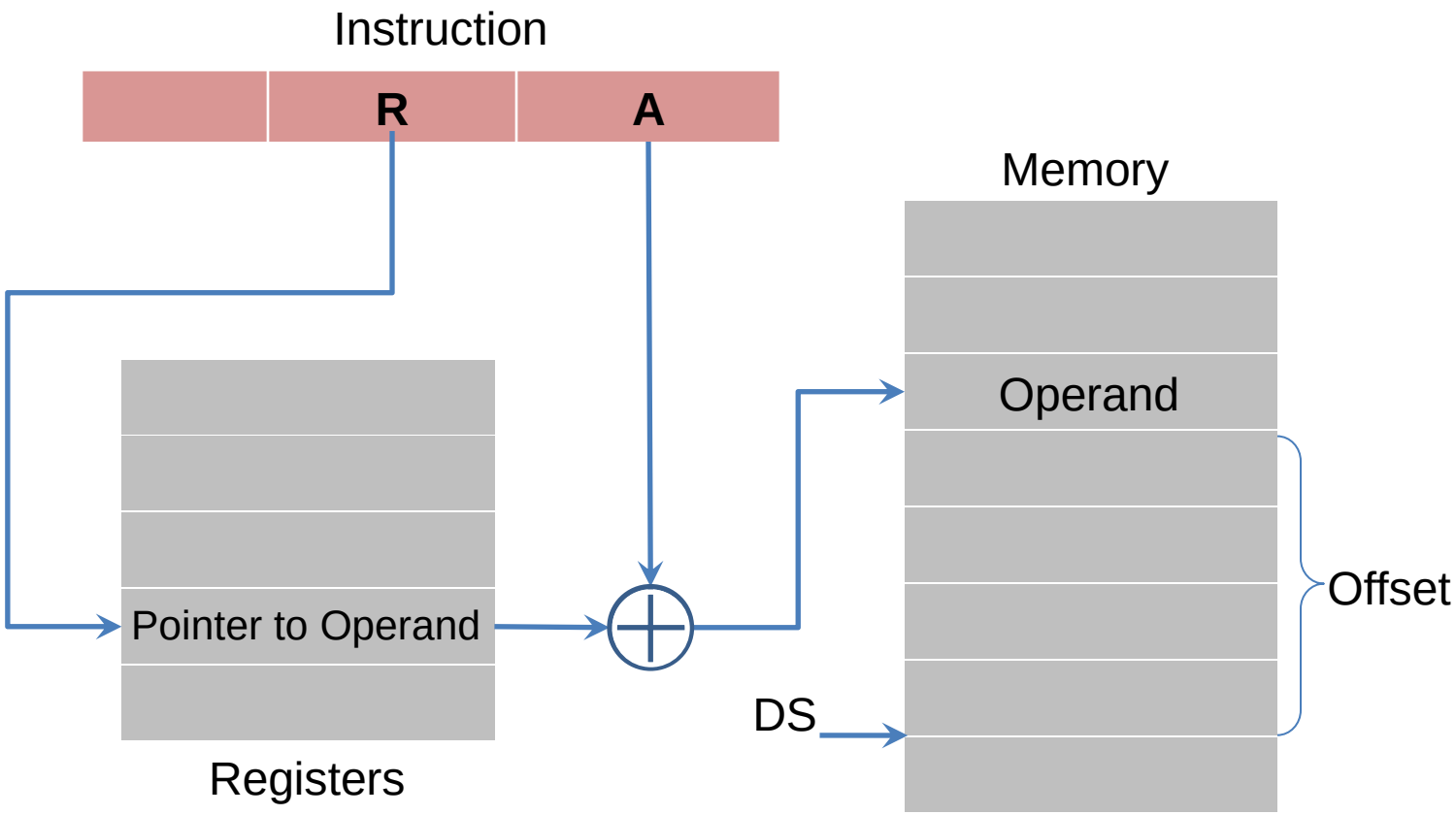
```
.data
    val DB 10h
.code
    MOV si, OFFSET val
    MOV al, [si]
```

Displacement Addressing (1/2)

- In this addressing mode, an instruction has two address fields
- Both these values are added to produce the effective address
- Displacement addressing can be used as
  - Relative Addressing
  - Base-Register Addressing
  - Indexing

$$EA = A + [R]$$

Displacement Addressing (2/2)



Relative Addressing

- Address of next instruction is added to address field to produce EA
- Address of next instruction is got from PC

$$EA = A + [PC]$$

Base-Register Addressing

- Register contains the main memory address
- Address field contains a displacement from address contained in register
- Examples are Segment Registers in x86

$$EA = A + R$$

Indexing

- Address field contains the main memory address
- Register contains the displacement from the address contained in address field

$$EA = A + R$$

Stack Addressing

- Operand is on the top of stack
- Operand can be pushed on the stack when it is necessary to save it
- Whenever operand is needed, it can be popped from the stack

Extract 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 reference
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