Tutorial #1
Addressing Modes

Anirban Lahiri & Prashant Agrawal
Department of Computer Science & Engineering
IIT Kharagpur

Outline

- Registers
- Addressing Modes

General Purpose Registers (GPRs)

- EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP
- 32-bit registers
- These are used as
  - Operands for logical and arithmetic operations
  - Operands for address calculations
  - Memory pointers

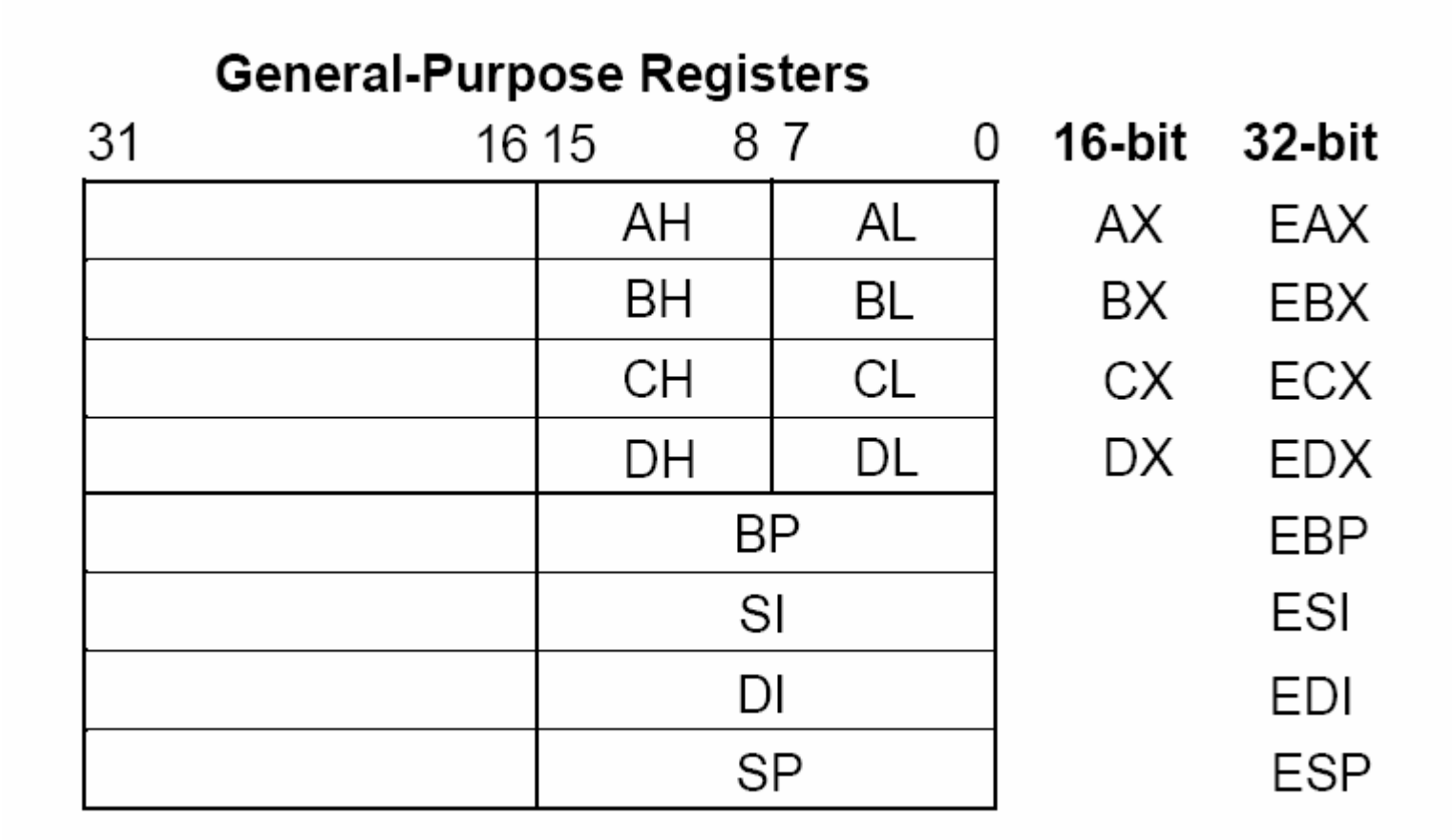
General Purpose Registers (GPRs)

- EAX
  - Accumulator for operands and result data
- EBX
  - Pointer to data in the DS segment
- ECX
  - Counter for string and loop operations
- EDX
  - I/O Pointer
- ESI
  - Source pointer for string operations

General Purpose Registers (GPRs)

- EDI
  - Destination pointer for string operations
- ESP
  - Stack pointer
- EBP
  - Pointer to data on the stack

General Purpose Registers



Ref: IA-32 Intel® Architecture Software Developer's Manual Volume 1: Basic Architecture

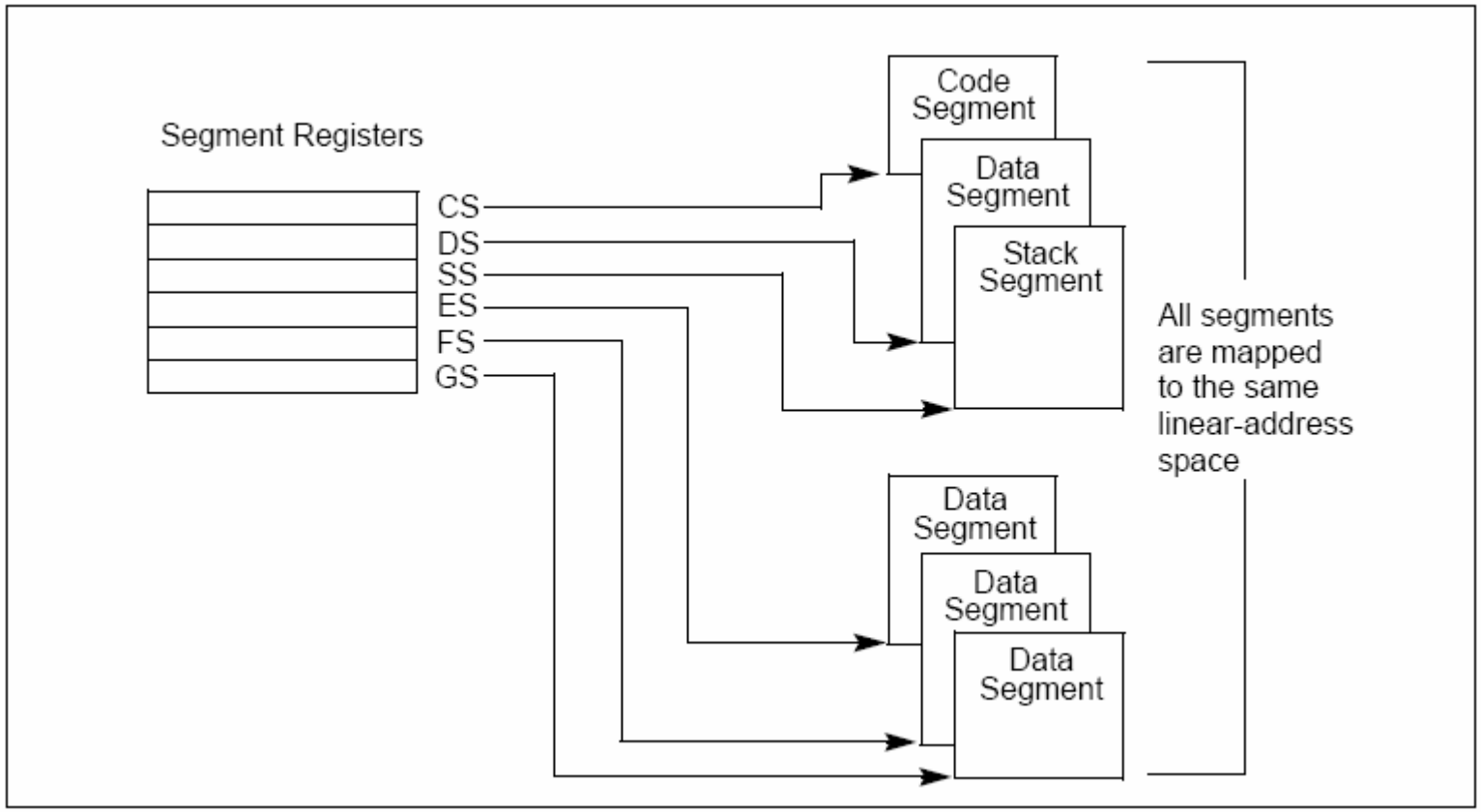
Segment Registers

- CS, DS, SS, ES, FS, GS
- These are 16-bit segment selectors
  - A segment selector is a special pointer that identifies a segment in memory
- CS
  - By default Instruction address in the code segment
- DS
  - By default Data address in data segment
- SS
  - By default Stack address in stack segment

Segment Registers

- ES
  - By default String destination address in data segment
- FS
  - No default
- GS
  - No default

Segment Registers



Ref: IA-32 Intel® Architecture Software Developer's Manual Volume 1: Basic Architecture

Outline

- Registers
- Addressing Modes

Addressing Modes

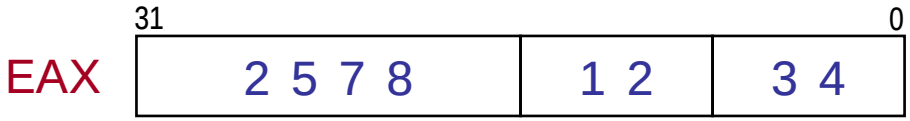
- Addressing mode of an operand determines how the processor interprets the operand address specified
- Common addressing modes
  - Immediate
  - Register
  - Direct
  - Register indirect
  - Memory indirect
  - Implied / Inherent
  - Auto Increment/Decrement
  - Base-Index
  - Relative / Displacement

Immediate Addressing Mode

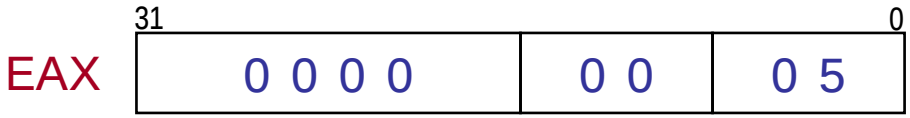
- The operand (data) is specified explicitly in the instruction itself

  Eg. mov $5, %eax

Before execution of the instruction



After execution of the instruction

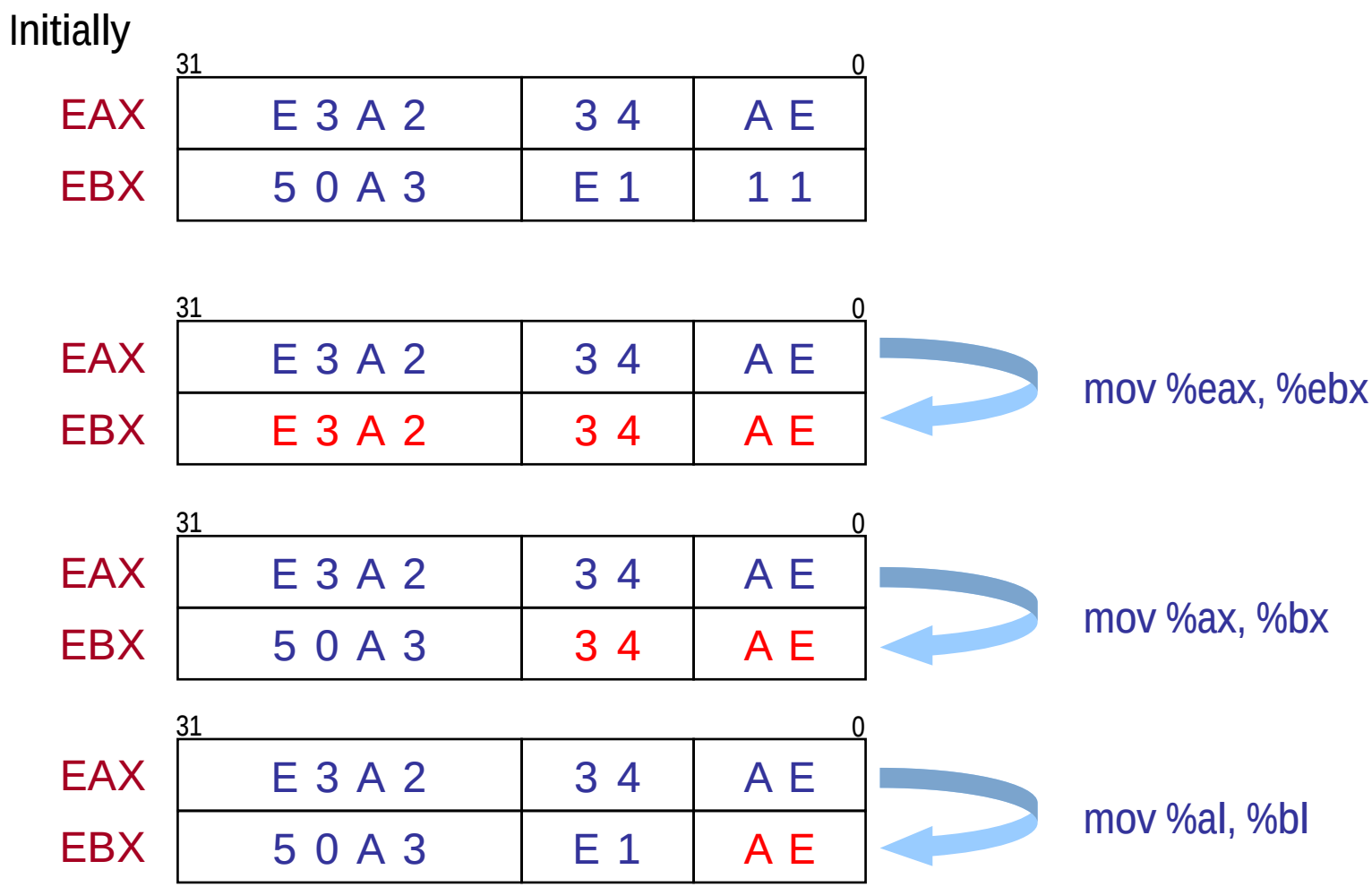


Register Addressing Mode

- The operands are contained in registers, which are specified by the instruction.
  - Eg. mov %eax, %ebx
- registers can be 8,16 or 32bits long
  - Eg.
    - mov %al, %bl #8-bit operands
    - mov %ax, %bx #16-bit operands
    - mov %eax, %ebx #32-bit operands

  Note: registers specified in a single instruction should be of the same size

Register Addressing Mode



Direct Addressing Mode

- The memory location of the operand is encoded in the instruction
  - Eg. mov 0x1024,%al
- The address is assumed to be an offset in the data segment
- For Intel x86 the direct addressing mode is indistinguishable from displacement addressing mode

Direct Addressing Mode

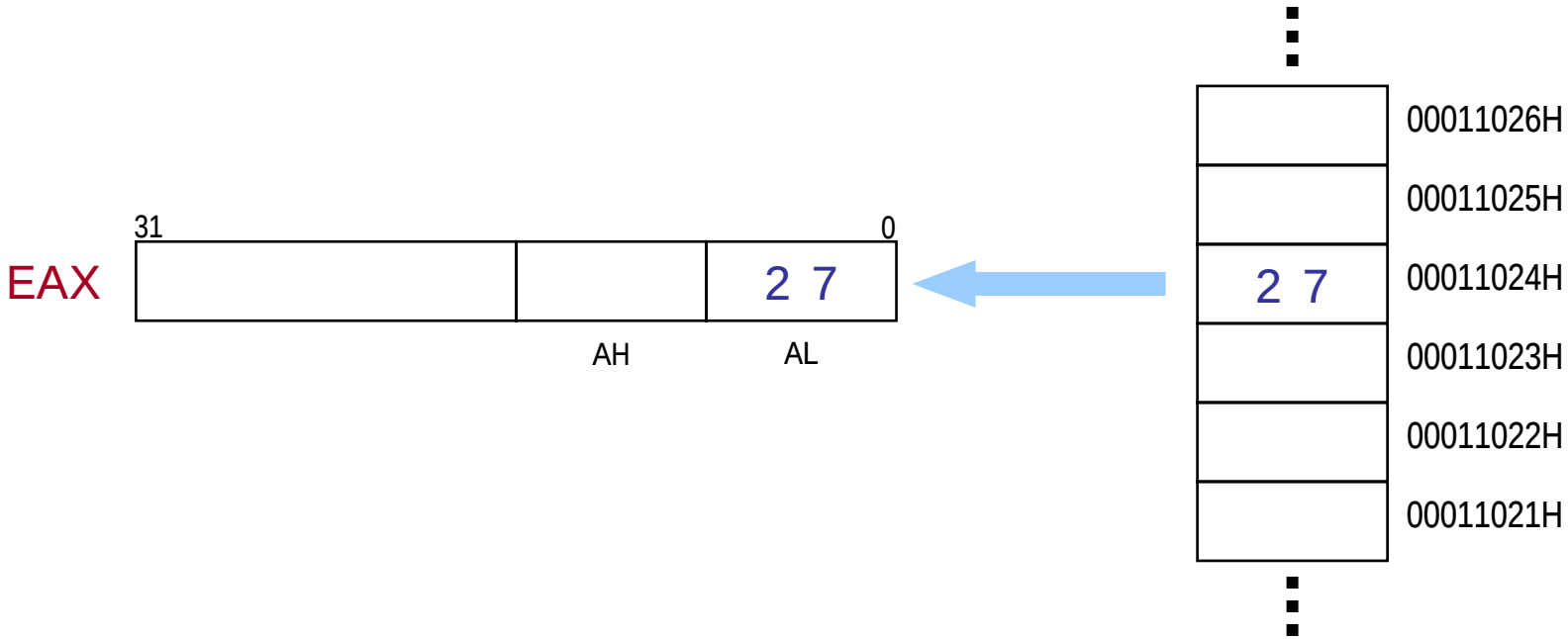
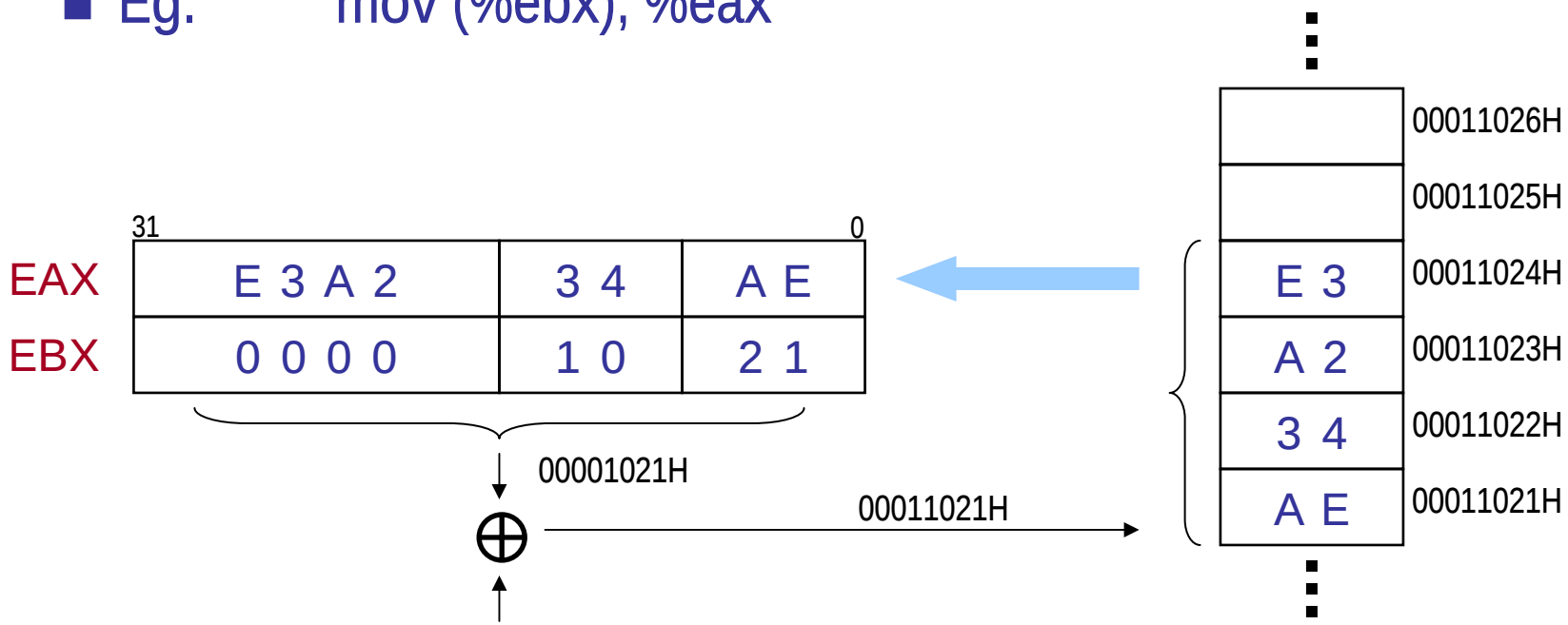


Fig 1.5: The operation of the mov (0x1024),%al instruction when base address of data segment is 00010000H

Register-Indirect Addressing Mode

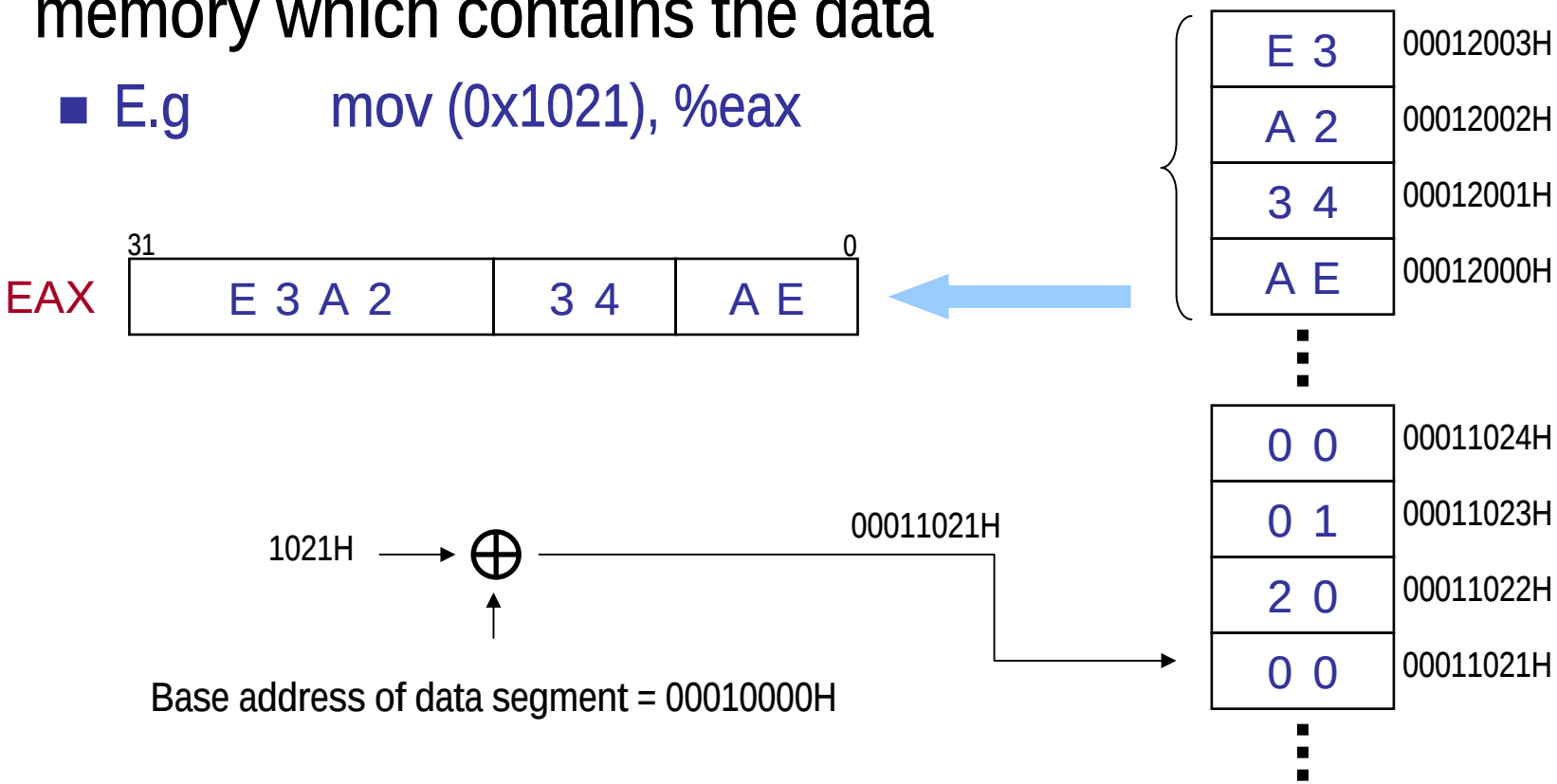
- The address of the operand is kept in a register. The register is indicated in the instruction
  - Eg. mov (%ebx), %eax



Base address of data segment = 00010000H

Memory Indirect Addressing Mode

- Address of location is specified in the instruction which is used as a pointer to another location in memory which contains the data
  - E.g mov (0x1021), %eax



Implied/ Inherent Addressing Mode

- Address of the operand is implied by the instruction rather than explicitly stated
  - E.g push %eax

Auto Increment/ Decrement Mode

- Designated register is used as a pointer to memory, and then the register is incremented/ decremented by the size of the operation
  - E.g.

```
mov $16, %ecx       # initializing counter
mov $0, %esi        # initializing source index
mov $160, %edi      # initializing destination index
cld                 # clear direction flag
rep                 # repeat
movs
```

Base-Index Addressing Mode

- It is an indirect addressing mode
- Uses one base register (BP or BX), and one index register (DI or SI) to indirectly address memory
- Base register holds the beginning location of a memory array
- Index register holds the relative position of an element in the array
  - E.g.

```
mov 0x10000, %ebx
mov $0, %edi
mov (%ebx, %edi), %eax
```

Scaled Base-Index Addressing Mode

- Similar to base-index addressing mode except for that the index register is multiplied by a scale
  - Eg. mov (%ebx,%esi,4), %eax

    Effective Address = (%ebx) + (esi)*4

Relative Addressing Mode

- It is similar to base-index addressing modes
- A displacement is added to the computed effective address.
- A variation of this mode is the relative scaled base-index addressing mode
  - Eg. mov -4(%ebx,%esi,4), %eax

    Effective Address = (%ebx) + (esi)*4 - 4

Note: section: disp (base, index, scale)

To be continued...