Assembly Language Programming

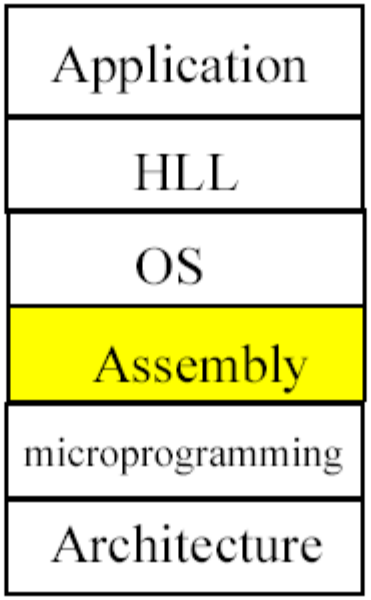
Assembly Programming

- Machine Language
  - binary
  - hexadecimal
  - machine code or object code
- Assembly Language
  - mnemonics
  - assembler
- High-Level Language
  - Pascal, Basic, C
  - compiler

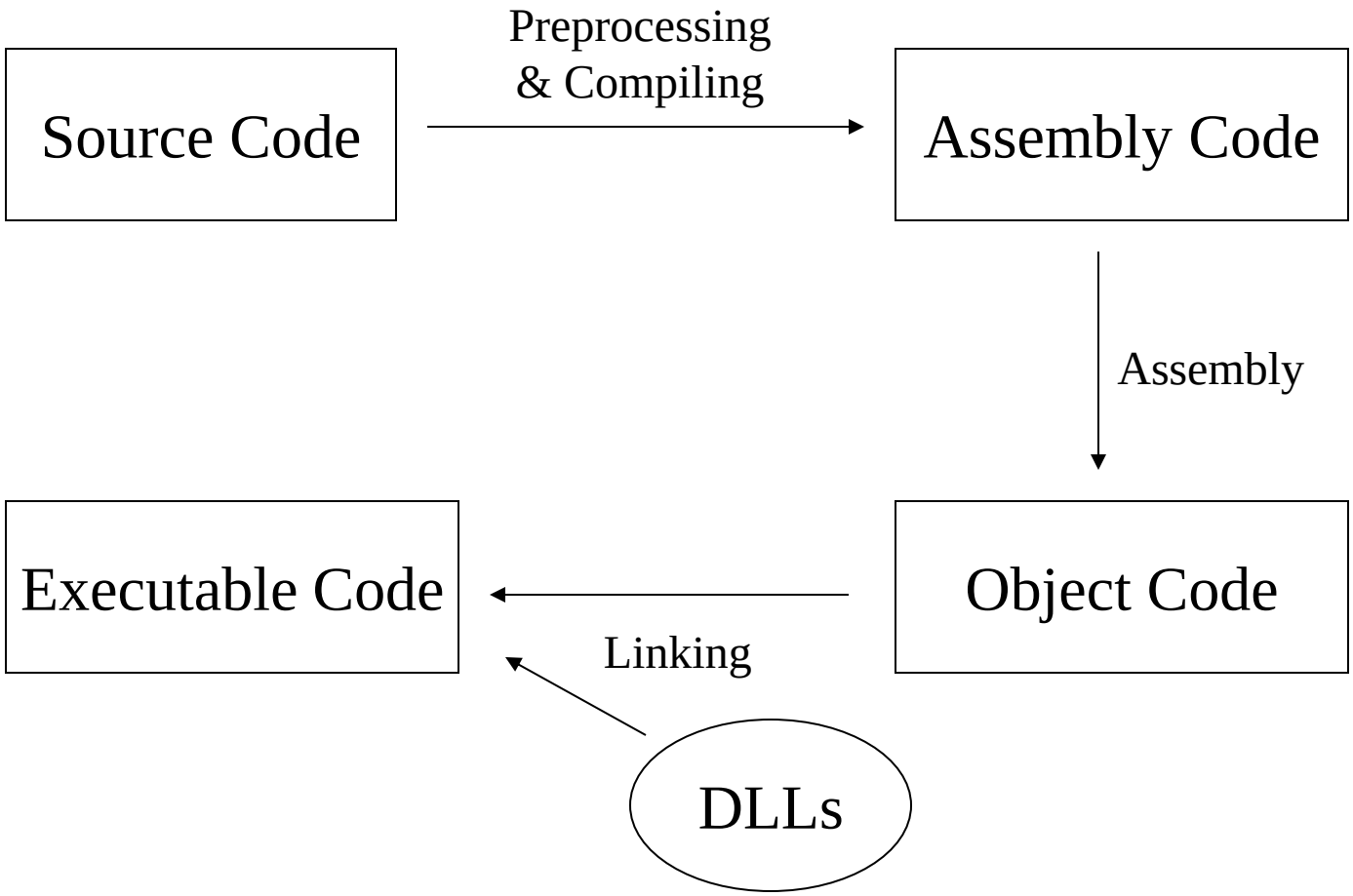
Assembly Language Programming

Motivations

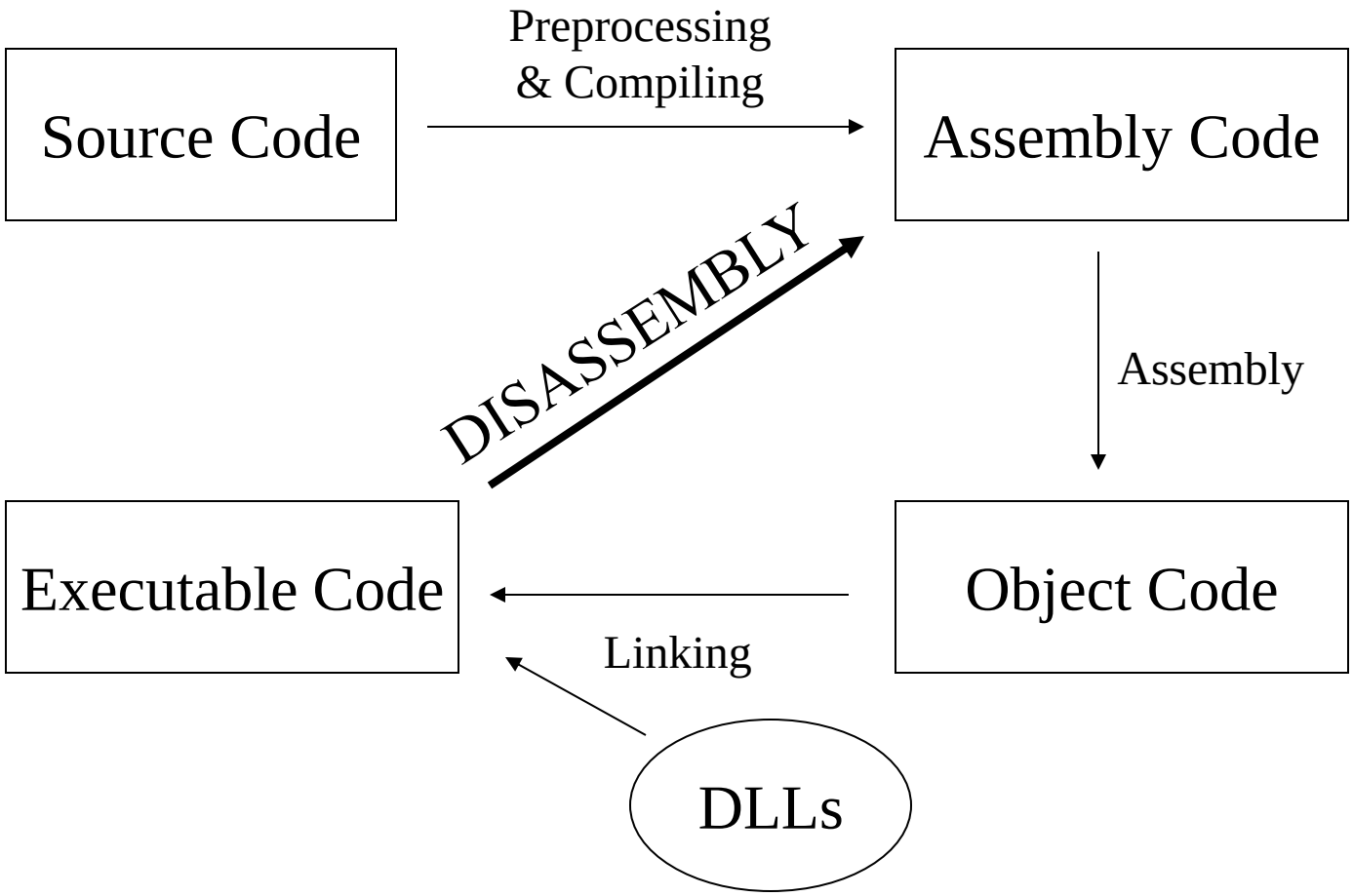
- Why do you learn assembly language?



What Does It Mean to Disassemble Code?



What Does It Mean to Disassemble Code?



Assembly Language

- Assembly language veya Assembler language
  - Bir bilgisayar veya diğer programlanabilen cihazlara yönelik bir alt-seviye programlama dilidir. **(low-level programming language)**
  - bu dilde genellikle mimarinin makine kod emirleri ile arasında güçlü bir (genellikle bire-bir) benzeşme vardır **(architecture's machine code instructions)**
  - (*Yüksek seviyeli programlama dillerinin tersine*) Her assembly dili, belirli bilgisayar mimarisine özgüdür.

High-level programming language

- High-level programming language
  - Genellikle çeşitli mimariler arasında taşınabilir **(portable)**
  - Fakat yorumlamaya veya derlemeye ihtiyaç duyar **(interpreting** or **compiling)**

Assembly Language

- Assembly language aynı zamanda
  - *Assembly*
  - *Assembler*
  - *ASM*
  - *Symbolic machine code*
  - *Assembly program*

  olarak da anılır.

Assembly Language

- Assembly dili, '*assembler*' olarak anılan bir utility program tarafından 'executable machine code' a dönüştürülür.

- Bu dönüşüm işlemi 'assembly' olarak anılır
  - *assembling the source code*

Assembly Language

- Semboller (*symbols*) tanımlayarak ve kullanarak donanımdaki bellek adreslerini (*memory adresses*) temsil etmeye imkan tanır
- ‘mnemonic’ olarak anılan özel sembol kullanılarak her bir alt-seviye makine emiri veya operasyon belirtilmiş olur.
- Tipik operasyonlar bir veya birden fazla operand gerektirir.

Assembly Programming

- Assembly dil emirleri 4 alandan oluşur

[label:] mnemonic [operands] [;comment]

- Label
- mnemonic, operands
  - MOV AX, 6764
- comment
  - ; this is a sample program

Model Definition

MODEL directive

- memory modelinin boyutunu seçer

- **MODEL MEDIUM**
  - Data must fit into 64KB
  - Code can exceed 64KB
- **MODEL COMPACT**
  - Data can exceed 64KB
  - Code cannot exceed 64KB
- **MODEL LARGE**
  - Data can exceed 64KB (but no single set of data should exceed 64KB)
  - Code can exceed 64KB
- **MODEL HUGE**
  - Data can exceed 64KB (data items i.e. arrays can exceed 64KB)
  - Code can exceed 64KB
- **MODEL TINY**
  - Data must fit into 64KB
  - Code must fit into 64KB
  - Used with COM files

Segments

- Segment definition:

The 80x86 CPU 4 tane segment registere sahiptir: CS, DS, SS, ES

- Segments of a program:

.STACK ; stack segmentin başlangıcına işaret eder

example:

```
.STACK    64    ; 64Byte'lık  yığın bellek alanı rezerve eder
```

.DATA ; data segmentin başlangıcına işaret eder

example:

```
.DATA1  DB    52H    ; DB direktifi bayt boyutunda parçalar halinde
                     bellek ayırır
```

.CODE ; Code segmentin başlangıcına işaret eder.

Assemble, Link, and Run Program

STEP	INPUT	PROGRAM	OUTPUT
1. Edit the program	keyboard	editor	myfile.asm
2. Assemble the program	myfile.asm	MASM or TASM	myfile.obj myfile.lst myfile.crf
3. Link the program	myfile.obj	LINK or TLINK	myfile.exe myfile.map

Assemble, Link, Run Files

STEP	INPUT	PROGRAM	OUTPUT

.asm – source file

.obj – machine language file

.lst – list file

- it lists all the Opcodes, Offset addresses, and errors that MASM detected

.crf – cross-reference file

- an alphabetical list of all symbols and labels used in the program as well as the program line numbers in which they are referenced

.map – map file

- to see the location and number of bytes used when there are many segments for code or data

Control Transfer Instructions

- NEAR – Kontrol, aktif code segment içerisindeki bir bellek lokasyonuna transfer edildiğinde

- FAR – Kontrol, aktif code segment dışında bir yere transfer edildiğinde

CS:IP – Bu register çifti her zaman sıradaki icra edilecek emirin adresine işaret eder

- NEAR Jump: IP güncellenir, CS aynı kalır
  (In a NEAR jump, IP is updated, CS remains the same)
- FAR Jump : hem CS hem de IP güncellenir
  (In a FAR jump, both CS and IP are updated)

Control Transfer Instructions

- Conditional Jumps
- Short Jump
  - Tüm koşullu dallanmalar kısa dallanmadır
  - Hedefin adresi, IP'nin –128 ile +127 byte aralığında olmalıdır
  - Koşullu dallanma 2-byte lık emirdir
    - Bir byte'ı dallanma şartının opcode'udur
    - 2.byte ise 00 ile FF arasında bir sayıdır
      - 256 olası addres:
        - forward jump to +127
        - backward jump to –128

Data Types and Data Definition

- Assembler data directives
  - **ORG** (origin) – to indicate the beginning of the offset address
    - *example:*

```
        ORG     0010H
```

  - **DB** (define byte) – allocation of memory in byte-sized chunks
    - *example:*

```
DATA1   DB      25          ;decimal
DATA2   DB      10001001B   ;binary
DATA3   DB      12H         ;hex
DATA4   DB      '2591'      ;ASCII numbers
DATA5   DB      ?           ;set aside a byte
DATA6   DB      'Hello'     ;ASCII characters
DATA7   DB      "O' Hi"     ;ASCII characters
```

Data Types and Data Definition

- Assembler data directives

  - **DUP** (duplicate) – to duplicate a given number of characters
    - *example:*

```
DATA1    DB     0FFH, 0FFH, 0FFH, 0FFH        ;fill 4 bytes with FF
```

Can be replaced with:

```
DATA2    DB     4 DUP(0FFH)                   ;fill 4 bytes with FF

DATA3    DB     30 DUP(?)                     ;set aside 30 bytes

DATA4    DB     5 DUP (2 DUP (99))            ;fill 10 bytes with 99
```

Data Types and Data Definition

- Assembler data directives

  - **DW** (define word) – allocate memory 2 bytes (one word) at a time
    - *example:*

```
DATA1    DW    342                ;decimal
DATA2    DW    01010001001B       ;binary
DATA3    DW    123FH              ;hex
DATA4    DW    9,6,0CH, 0111B,'Hi'  ;Data numbers
DATA5    DW    8 DUP (?)          ;set aside 8 words
```

  - **EQU** (equate) – define a constant without occupying a memory location
    - *example:*

```
COUNT    EQU    25

;COUNT can be used in many places in the program
```

EXE vs. COM

- COM files
  - Smaller in size (max of 64KB)
  - Does not have header block

- EXE files
  - Unlimited size
  - Do have header block (512 bytes of memory, contains information such as size, address location in memory, stack address)