Assembly Programming

Karol Regula

What is Assembly? - Brief History

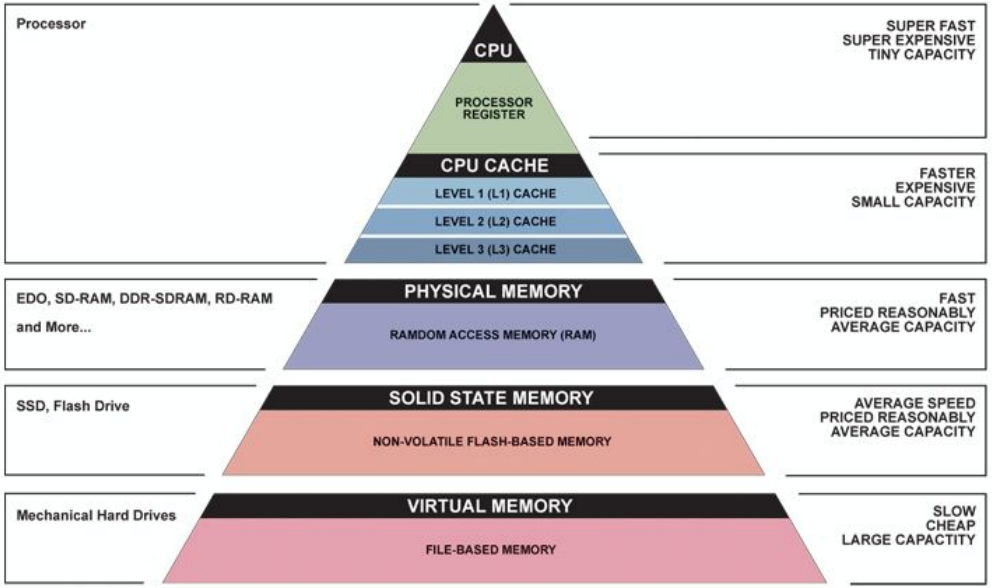
- Very low level programming language
- Streamlined way of writing machine code
- **One to one** correspondence between **machine instructions** and assembly code
- Device (processor architecture) specific
  - pre-defined instruction sets
- Translated into machine language by **assembler**

Terminology

Register - easily accessible piece of memory on processor die

Accumulator - special register in which calculation results are stored (used in older processors, intel still stores multiplication in EAX and EDX registers)

32-bit Intel CPUs have 8 registers (EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI), 64-bit CPUs have 16



Simple Example

Load register AL with following byte of code: 01100001

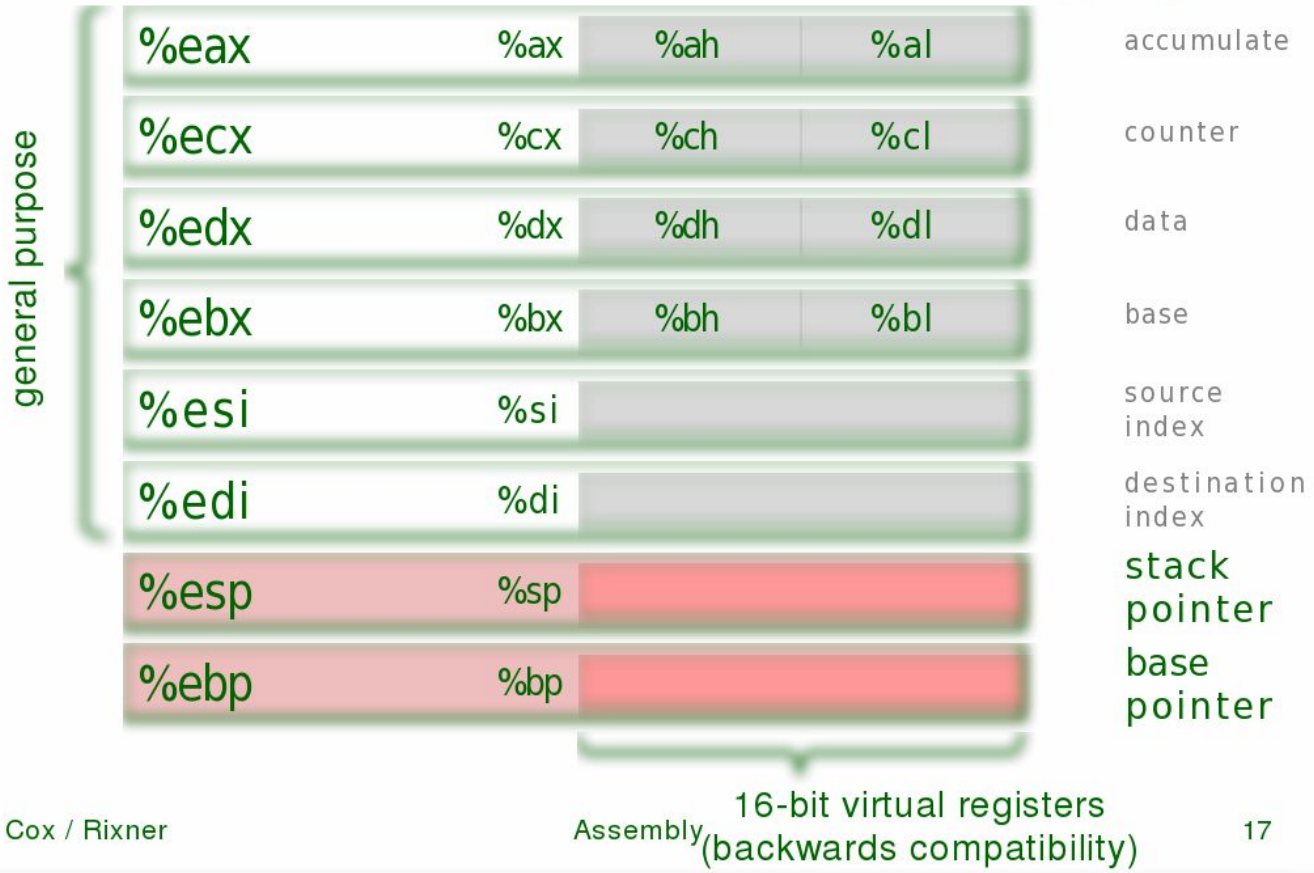
Binary: 10110000 01100001

Hexadecimal: B0 61

Assembly: MOV AL, 61h

Opcode - CPU instruction

Integer Registers (IA32)



Usage

- Compiled languages are eventually converted into assembly
- Wirth’s law - software is becoming slower at a faster rate than hardware is becoming faster
- Compilers usually have options to generate assembly code

- Low level applications, **interrupt handlers**, and hardware drivers benefit from the efficiency of assembly
- “Getting money’s worth” out of legacy hardware
- Firmware for appliances
- Situations where precise timing is required (airplanes, medical equipment)
- High security programs

Advantages of Assembly

- Much higher level of control -> better optimization & performance

Usage in the Game Boy (More Complex Example)

Updating life counter once score exceeds 100:

```
score  = 95
life = 1

...
score += 1

if score > 100:
        score = 0
        life += 1
```

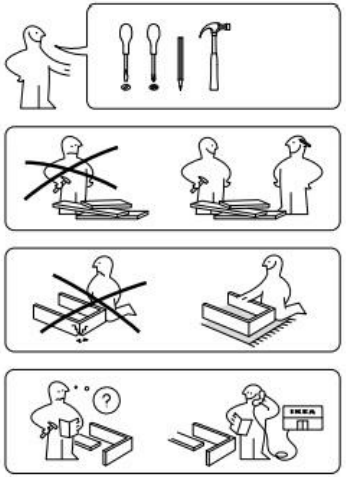
```
LD A, 0
LD (8000H), A ; store score in memory
…
LD A, (8000H)       ; load score from memory, commence check
INC A
LD (8000H), A
SUB 100             ; from accumulator
JP C, AFTER         ; jump to AFTER if previous operation is negative
LD A, 0
LD (8000H), A
LD A, (8001H)
INC A
LD (8001H), A

AFTER:  ;  when score is not above 100 (move along)
```

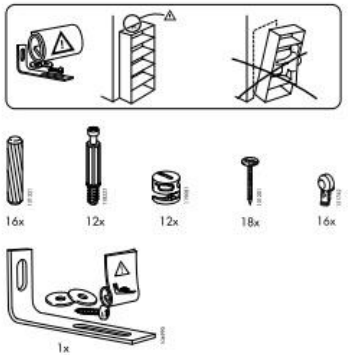
Relevance of Assembly Today

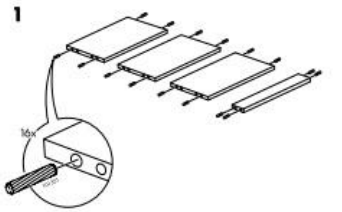
TIOBE index of programming language popularity

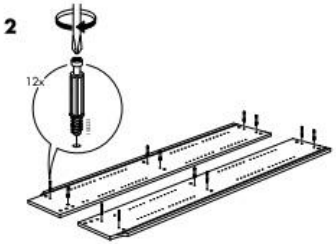
https://www.tiobe.com/tiobe-index/

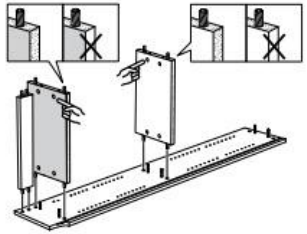
Mar 2017	Mar 2016	Change	Programming Language	Ratings	Change
1	1		Java	16.384%	-4.14%
2	2		C	7.742%	-6.86%
3	3		C++	5.184%	-1.54%
4	4		C#	4.409%	+0.14%
5	5		Python	3.919%	-0.34%
6	7	^	Visual Basic .NET	3.174%	+0.61%
7	6	v	PHP	3.009%	+0.24%
8	8		JavaScript	2.667%	+0.33%
9	11	^	Delphi/Object Pascal	2.544%	+0.54%
10	14	^^	Swift	2.268%	+0.68%
11	9	v	Perl	2.261%	+0.01%
12	10	v	Ruby	2.254%	+0.02%
13	12	v	Assembly language	2.232%	+0.39%
14	16	^	R	2.016%	+0.73%
15	13	v	Visual Basic	2.008%	+0.33%
16	15	v	Objective-C	1.997%	+0.54%
17	48	^^	Go	1.982%	+1.78%
18	18		MATLAB	1.854%	+0.66%
19	19		PL/SQL	1.672%	+0.48%
20	26	^^	Scratch	1.472%	+0.70%

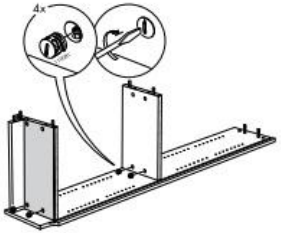


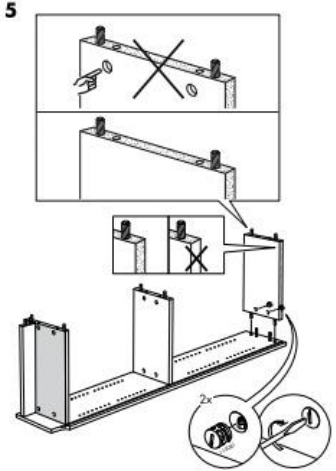

16x
12x
12x
18x
16x
1x

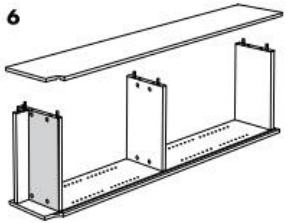

1
16x

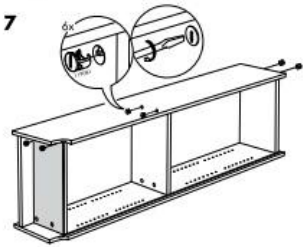

2
12x

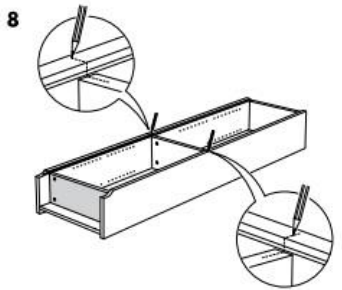

3

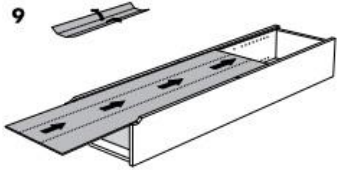

4
4x


5
2x


6


7
6x


8


9

```
global _start
section .data

        hello    db      "Hello, World!", 10
        length   equ     $-hello

section .text
_start:

        mov eax, 4              ; write to file
        mov ebx, 1              ; STDOUT handle
        mov ecx, hello          ; message
        mov edx, length         ; size of message

        int 80h                 ; execute the syscall

        xor ebx, ebx            ; send 0 as 'exit code'
        mov eax, 1              ; terminate process
        int 80h                 ; execute the syscall

        ;; intel notation is used (as opposed ot AT&T)
```

```
global _start

section .data

        hello       db   "Hello, World!", 10

        length      equ $-hello

section .text

_start:

        mov eax, 4              ; write to file

        mov ebx, 1              ; STDOUT handle
```