

Archi2 Gyak

(Processzorok Utasításszintű Kezelése)

2014 ősz

Ajánlott irodalom

- **Agárdi Gábor: Gyakorlati Assembly, LSI Oktatóközpont, 1996, ISBN 963 577 117 7**
- **Agárdi G.: Gyakorlati Assembly haladóknak, LSI oktatóközpont, 1996**
<http://fotoblog.digilab.hu/blog/modern-idoutazas/>
- **Dr. Gidófalvi Z.: Programozás MASM Assembly nyelven, Műegyetemi kiadó, 1995**
- **MOODLE (elearning.uni-obuda.hu)**
- **mobil.nik.uni-obuda.hu**

Számonkérések

10. alkalom	Elméleti ZH (nov 24-27)
11. alkalom	Gyakorlati ZH (dec 1-4)
12. alkalom (utolsó hét)	Pótlás/Javítás (dec 8-11)
Vizsgaidőszak	Gyakjegypót (dec 15-18)

A hallgató abban az esetben kaphat aláírást, ha hiányzása a gyakorlati órákról nem haladja meg a TVSZ-ben meghatározott 30%-ot.

A 12. laboralkalom során csak az elégtelen értékelést kapott és a ZH-t nem írt hallgatók pótolhatnak a teljes anyagból.

Követelményrendszer:

Az Elméleti és a Gyakorlati ZH-n is egyenként el kell érni a min. 50%-ot az aláíráshoz!

Elért eredmény

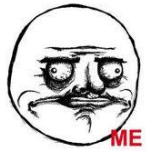
50%-100%

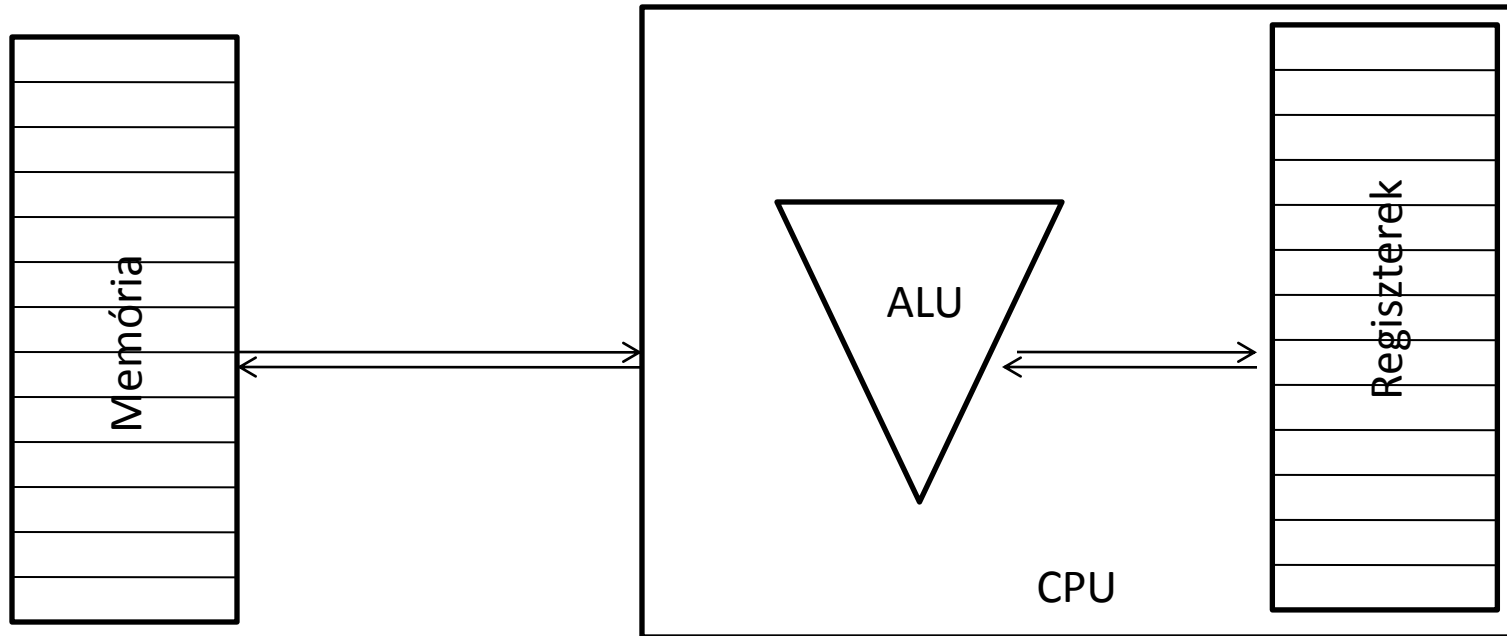
0%-50<%

Aláírva

Aláírás megtagadva

Számítógép felépítése

(*Archi 1* )



Számítógép felépítése

- CPU: memóriában tárolt adatokat kiolvassa, értelmezi és végrehajtja
- ALU: műveletvégző
- Regiszterek: ~tárolók, sokkal gyorsabb, mint a memória közvetlen hozzáférése
- RAM: elsődleges adattároló (bit/byte/word/dword)
 - Memóriaszervezés byte alapú (minden byte-hoz egy cím van rendelve)

Gépi kód

Műveleti kód	Operandus
--------------	-----------

- Számkódok (utasítás, érték)
- Műveleti kód (utasítás): MIT kell csinálni
- Operandus (érték): HOL/MIN kell végrehajtani
- Gépi kód helyett jelképes nevek (mnemonic-ok)
 - Pl.: MOV (move), ADD, MUL (multiply)

Assembly kód

Műveleti kód	Operandus
--------------	-----------

- Pl.: Assembly: MOV CX, 4095
 Gépi kód: B9h 0F FFh
- Az assembly kód írásakor a mnemonikokat használjuk, hogy a gépi kódú utasításokat kifejezzük
- *Használhatók direktívák, címkék és deklarálhatók változók is*

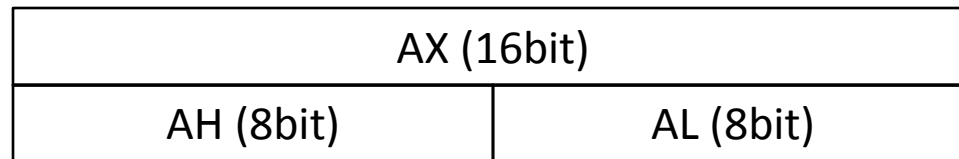
Regiszterek

(8086 processzor)

- Általános célú regiszterek (16 bites):

AX, BX, CX, DX

Alsó, felső részre osztható (2*8bit)
Regiszter



Külön címezhető felső alsó RÉSZ ... nem önálló regiszter!!

Decimális	Bináris		Hexadecimális
AX	AH	AL	AH AL
46	00000000	00101110 b	00 2E h
5	00000000	00000101 b	00 05 h
65535	11111111	11111111 b	0FF FF h

Regiszterek

(8086 processzor)

- AX: számos aritmetikai utasítás cél, ill. forrásregisztere
- BX: memóriacímzésnél használható bázisnak
- CX: ciklusokhoz használható, minden iteráció során eggyel csökken
- DX: I/O utasítások itt tárolják a port számaikat, ill. bizonyos aritmetikai utasítások használják

Regiszterek

(8086 processzor)

- Dedikált regiszterek
 - SP: veremmutató *(a verem tetejére mutat)*
 - BP: bázismutató *(a verem egy elemét jelöli ki)*
 - SI: forrásindex *(string kezelő utasítások használatakor a forráscímet tartalmazza)*
 - DI: célindex *(cél string címét tartalmazza)*
 - SR: státusz regiszter (Flag)
- 16 bites címmel megcímezhető legnagyobb memória értéke 65535 (64K memória) Nagyobb megcímezhető memória használata érdekében:
 - CS: kód szegmens
 - DS: adat szegmens
 - ES: extra szegmens
 - SS: verem szegmens

Regiszterek

(8086 processzor)

- Memória ~ könyv
- Oldal ~ szegmens regiszter
- Oldal sora ~ index(offset) regiszter
- Szegmensek egymástól 16byte távolságra vannak
- Egy címet 20 biten ábrázolhatunk (átfedés a szegmensek között)
- Lineáris cím = Szegmens cím : Offset cím

Lineáris cím meghatározása:

1. a szegmenst eltoljuk balra 4 bittel

2. az offszetet ehhez hozzáadjuk

pl.: szegmens: 8000h

 offszet: 1234h

$$\begin{array}{r} 1234 \\ + 80000 \\ \hline 81234h \end{array}$$

SR - Statusz regiszter (Flags)

(read only!!! 😊)

a Flageket tartalmazza, amelyek az állapotteret reprezentálják, ezek bizonyos műveletek hatására állapotot válthatnak kivonás, túlcsordulás, stb.)

C (CARRY FLAG – átviteljelző bit)

1-re áll, ha az aritmetikai művelet során átvitel vagy áthozat keletkezik bitléptető és forgató utasítások során is használatba kerül

P (PARITY FLAG – paritásjelző bit)

adatkommunikációs alkalmazások során szükséges páros paritás esetén értéke 0

A (AUXILIARY CARRY FLAG – segédátvitel jelző bit)

BCD (binárisan kódolt decimális) aritmetikában használatos

Z (ZERO FLAG – zérusjelző bit)

1-re áll ha az eredmény zérus

S (SIGN FLAG – eljelző bit)

negatív eredmény esetén 1-re áll

O (OVERFLOW FLAG – túlcsordulás jelző bit)

egy matematikai művelet eredménye meghaladja a kiszabott tartományt, akkor 1-re áll

Regiszterek

(80386 processzor kiegészítések)

- regiszterek
 - EAX, EBX, ECX, EDX (32bites)
 - ESP veremmutató, EBP bázismutató
 - ESI (forrásindex), EDI (célindex)
- szegmens regiszterek
 - CS (code), DS (data), SS (stack), ES (extra), FS, GS (16bites)

EAX (32bit)		
<i>Külön nem címezhető</i>	AX (16bit)	
<i>Külön nem címezhető</i>	AH (8bit)	AL (8bit)

Lehetséges címzési módok

- MOV AH, 0 AH-ba 0-t tölt
- MOV AL, 04 AL-be 4-et tölt
- MOV AX, 42134 AX-be 42134-et tölt
- MOV AX, BX AX-be betölti BX tartalmát
- MOV AH, BL AH-ba betölti BL tartalmát (*azonos regiszterméret!*)
- MOV AX, 0FFFFh AX-be betölti 0FFFF hexa számot (65535 decimális, *a betűvel kezdődő hexadecimális szám elé 0-t kell írni!*)
- MOV AL,-40 AL-be betölti -40 decimális számot
- MOV AX,OFFSET MYDATA
 A MYDATA szegmensben belüli „offset” címe kerül AX-be
 MYDATA címke egy memóriacímke, ahol az adat van
- MOV AX,[SI] AX-be tölti az SI regiszter által mutatott 16 bites értéket
- MOV AL,[SI] AL-be tölti az SI regiszter által mutatott 8 bites értéket
- MOV AX,[SI+2] Az SI regiszter+2 cím által mutatott adat kerül AX-be
-

Megszakítások

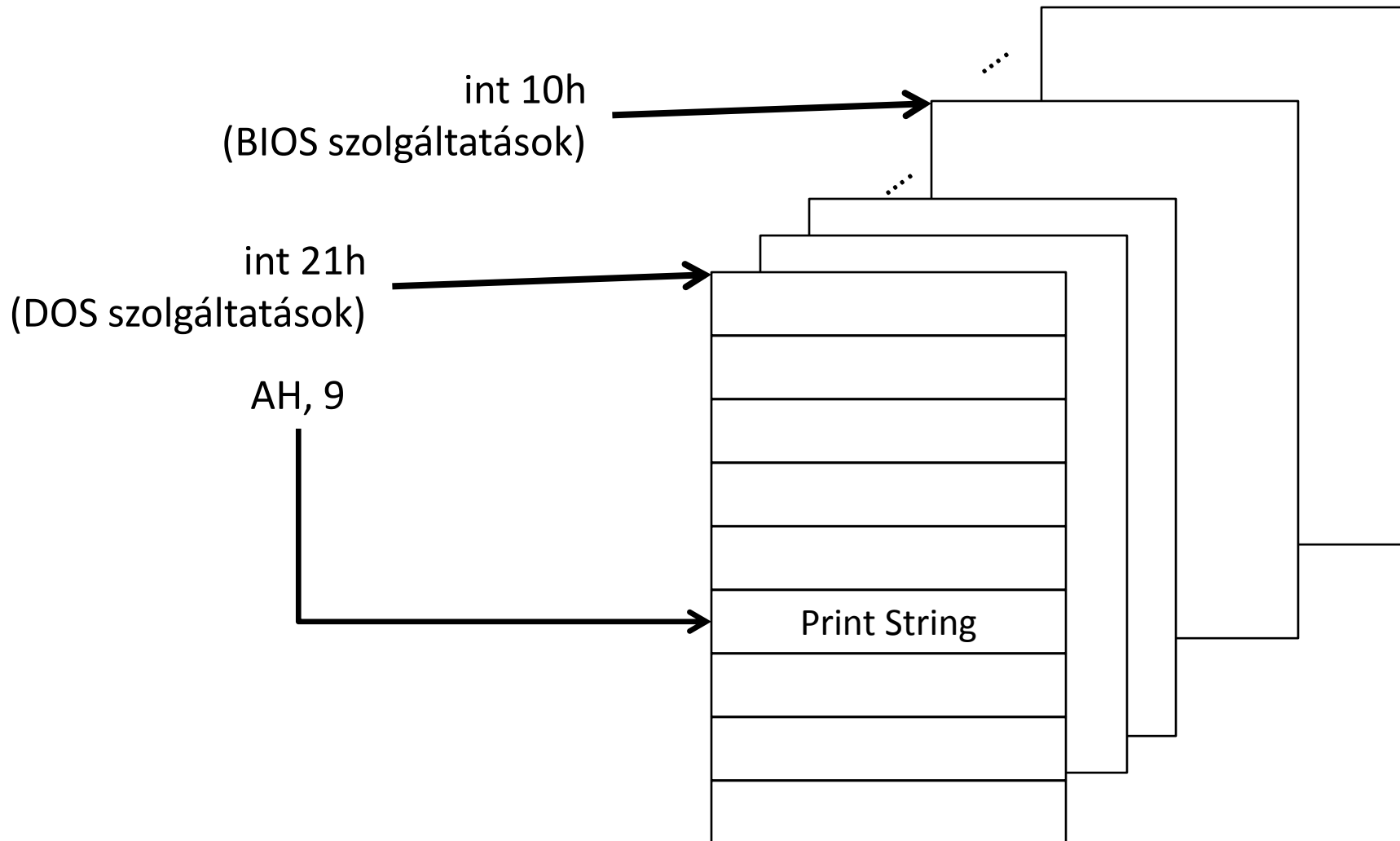
- Hardver megszakítások
- Szoftver megszakítások
 - a program kódjából kerül aktiválásra
 - a program futásával szinkron, mindig ugyanott következik be (INT utasítások)
 - nem maszkolhatóak

Szoftver megszakítások

- amikor egy szoftvermegszakítás meghívásra kerül, az INT utasítás után megadott azonosítóhoz tartozó kiszolgáló rutin fut le
- ebből 256 db van: INT 00h...INT 0FFh
- a megszakítás-kezelő rutinok kezdőcímei a megszakítás vektortáblában található
- ezt a „táblázatot” az operációs rendszer tölti be a bootolási folyamat során
- az INT utasítások ebből olvassák ki a szolgáltatások kezdőcímét
- INT n hatására az n. bejegyzés tartalma lesz a következő végrehajtandó utasítás címe

INT 21H

- az int 21h-t szokás MS-DOS API-nak is hívni, mivel a legtöbb operációs rendszeri funkciót ezen keresztül tudjuk elérni
- azt, hogy a szolgáltatásnak melyik al-funkcióját akarjuk használni, regiszterekben közöljük (pl. AH-ban)
- Pl. karakter beolvasás, karakter kiírás, string kiírás



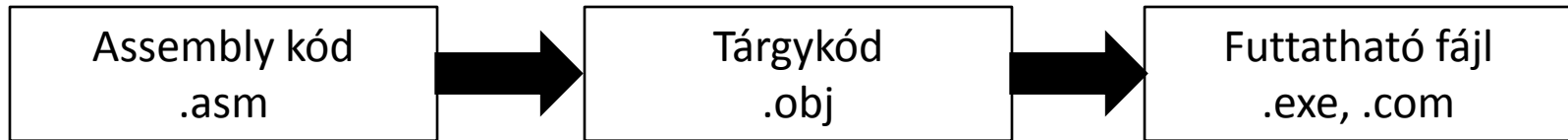
Gyakori utasítások áttekintése

- **MOV** – adatmozgatás
- **ADD** – összeadás
- **SUB** – kivonás
- **CMP** – összehasonlítás
- **JMP** – feltétel nélküli vezérlés átadás
- **JZ, JNZ, JC, JNC, ...** - feltételes vezérlést átadó utasítások
- **PUSH, POP** – vermet kezel utasítások
- **INT** – megszakítási eljárás hívása
- **CALL, RET** – szubrutin hívás

COM vs. EXE

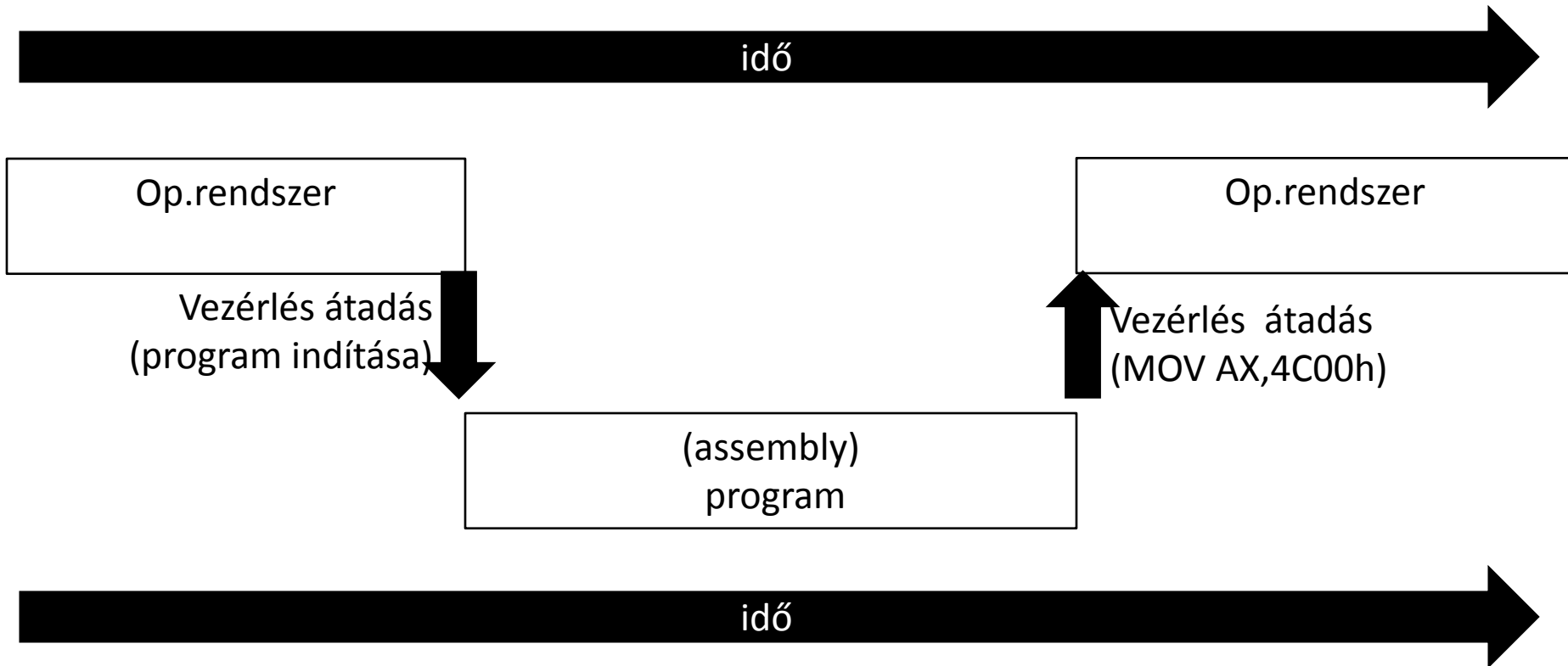
- COM: a teljes program egy szegmensben van - 64K (*CP/M*)
 - Nincs fejlécük (header)
 - Csak tisztán a futtatható kódot és a hozzáírt adatokat tartalmazzák
- EXE: külön kód, adat és veremszegmens, mindegyik lehet 64K (*DOS*)
 - Van fejlécük (header)
 - A fejléc alapján a DOS a programban szerepl címekeket betöltéskor át tudja írni, pl.: az adatszegmens címe

Fordítási folyamat



- Az assembly kódot az **Assembler** fordítja tárgykódra
 - Fordítóprogram, amely tárgykódot generál
 - Pl.: MASM (Microsoft), TASM (Borland)
 - Ezeket az operációs rendszer még nem tudja futtatni!
- A gépi kódot tartalmazó állományból a **Linker** segítségével készíthetünk COM ill. EXE fájlokat
 - Olyan program, amely a tárgykódból futtatható fájlt generál az operációs rendszer számára
 - Pl.: LINK, TLINK

Program futtatása



Assembly kód vázlat

<i>Kód</i>	Segment		;kód szegmens kezdete
	assume	CS:Kód, DS: Adat, SS:Stack	;szegmensek definíciója
<i>Start.</i>	.		;program kezdetét jelző címke
	.	;programkód helye	
	.		
<i>Kód</i>	Ends		;kód szegmens vége
<i>Adat</i>	Segment		;adat szegmens kezdete
	.		
	.	;változók helye	
	.		
<i>Adat</i>	Ends		;adat szegmens vége
<i>Stack</i>	Segment		;stack szegmens kezdete
	.		
	.		
	.		
<i>Stack</i>	Ends		;stack szegmens vége
	End	<i>Start</i>	;címke vége

Első gépi kódú program (~Hello World!)

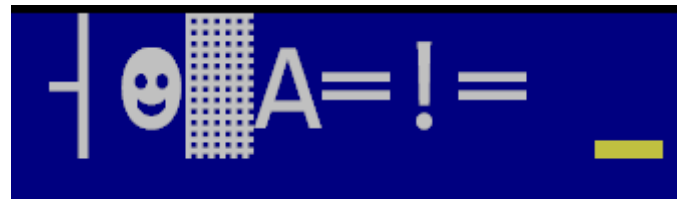
also.com program:

B4	02	B2	41	CD	21	CD	20	hex
180	2	178	65	205	33	205	32	dec

Alt+180

Alt+2

Stb...



Első gépi kódú program

- A program Assembly nyelvű megfelelője:

mov ah,2

mov dl,41h ;"A"

int 21h

int 20h

<i>Hex</i>	<i>dec</i>	<i>mnemonik</i>	<i>utasításkód</i>	<i>adat (operandus)</i>	<i>megjegyzés</i>
B4	02	mov	ah,	2	
B2	41	mov	dl,	41h	;"A"
CD	21	int	21	h	
CD	20	int	20	h	