# Digitális technika 2. jegyzet

Sebők Bence

2017. tavasz

## Contents

# 1 Bevezetés

Ez a jegyzet a BME Digitális technika 2. tantárgyhoz szeretne segítséget nyújtani. Ezen kezdeményezés célja, hogy segítsen a hallgatóknak megérteni a tananyagot. A tantárgy nehéz, erősen ajánlott előadásra, gyakorlatra járni. Ez a jegyzet csak az előadáson, gyakorlaton készült jegyzet mellé jelent segítséget, nem tanít meg a nulláról a tantárgy minden apróságára.

Ez egy hallgatói jegyzet, nincs lektorálva, se egyetemi oktató által felügyelve. Amit itt olvasol, azt csak saját felelősségre használd, hivatalos helyeken nem hivatkozási alap ez a jegyzet.

# 2 Alapismeretek

## 2.1 Intel 8085 processzor

- PC (Program Counter, program számláló): regisztárpár, ami az aktuálisan használt memóriacímet tartalmazza.
- SP (Stack Pointer, stack mutató): a stack tetejére mutató pointer (regiszterpárban tárolja ezt a 2 bájtos címet)

# 3 Assembly

Digitális technika 2. tantárgy során az Intel 8085 processzor programozásához használt Assembly programozási nyelvet tanítják. Ez a fejezet erről az Assemblyről szól.

## 3.1 Tanszéki szimulátor használata

Van egy az IIT-n fejlesztett Intel 8085 szimulátor. Ennek segítségével tetszőleges kódot tudunk futtatni egy virtuális Intel 8085 processzoron. Tetszőleges kód alatt értem, hogy bármit, ami a processzor utasításkészlete és fordítói direktívái lehetővé tesznek.

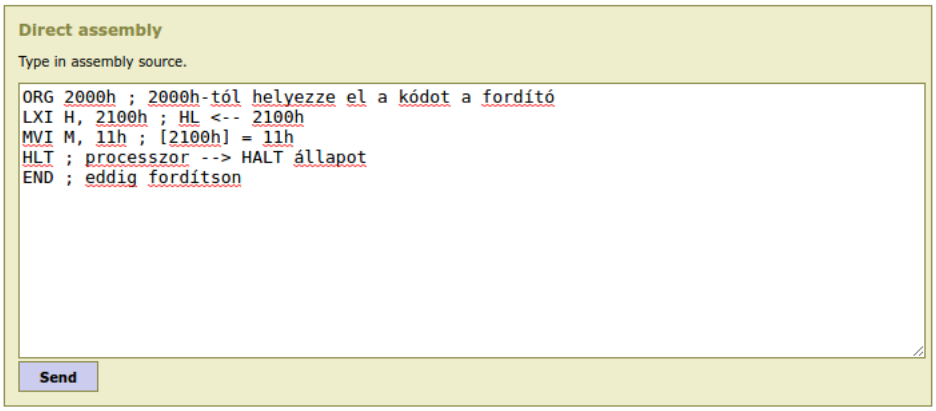
A tanszéki szimulátor a következő linken érhető el:
`http://topcat.iit.bme.hu/tools/i8085sim/i8085sim.cgi`
A szimulátor a következő módon néz ki:


A következő kódot szeretnénk a szimulátorban futtatni:

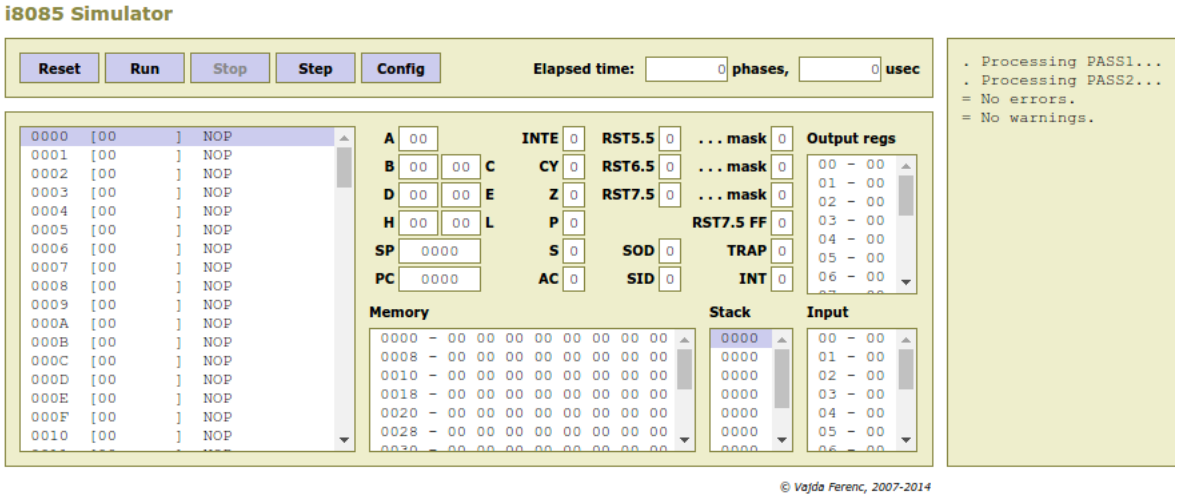
```
ORG 2000h ; 2000h-tol helyezze el a kodot a fordito
LXI H, 2100h ; HL <-- 2100h
MVI M, 11h ; [2100h] = 11h
HLT ; processzor --> HALT allapot
END ; eddig forditson
```

### 3.1.1 1. lépés: kód beillesztése

A szimulálandó kódot a Direct assembly ablakba kell írni:




Ezután a Send gombra kattintva menjünk tovább, ahol ez fogad minket:




### 3.1.2 2. lépés: szimuláció beállítása

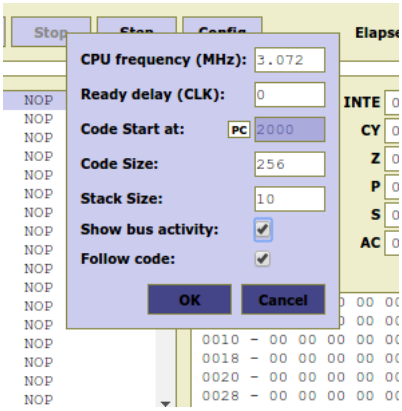
Néhány beállítást el kell végeznünk, hogy a megfelelő módon tudjuk nyomon követni a futó kódot.

1. Coda Start at: cím, ahova elhelyezzük a kódot (és kattintsunk bele a fehér téglalapba, hogy megjelenjen ott a PC felirat)

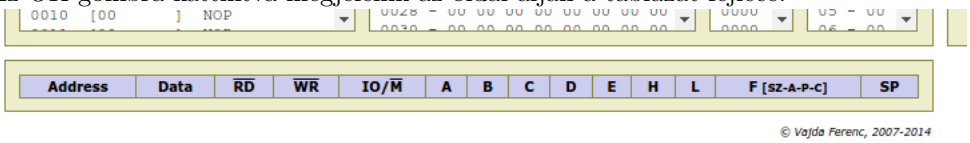
2. Show bus activity: legyen bepipálva

3. Follow code: legyen bepipálva

Az utolsó 2 opció ahhoz kell, hogy egy szép táblázatos formában jelenítse meg a kód futását és utasításonként haladjon a kód végrehajtása során. Ezek elvégzése után ezt kell látni:



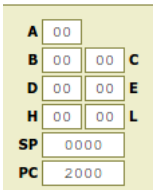

Az OK gombra kattintva megjelenik az oldal alján a táblázat fejléce:




A táblázat fejléce:

1. Address: milyen címre mutat éppen a PC
2. Data: milyen adat van ezen a memóriacímen
3. $\overline{RD}$: olvasás-e a jelenlegi művelet
4. $\overline{WR}$: írás-e a jelenlegi művelet
5. $IO/\overline{M}$: a mostani utasítás memória vagy periféria műveletet hajt végre
6. regiszterek aktuális értékei
7. F [SZ-A-P-C]: flag-ek aktuális értékei
   (a) Sign flag: előjel flag
   (b) Zero flag: zérus flag
   (c) Auxillary carry flag: fél-átvitel flag
   (d) Parity flag: paritás flag
   (e) Carry flag: átvitel flag
8. SP: Stack Pointer aktuális értéke

A PC mezőbe írjuk bele a címet, ahova helyeztük a kódot, jelen példában a 2000h-t:

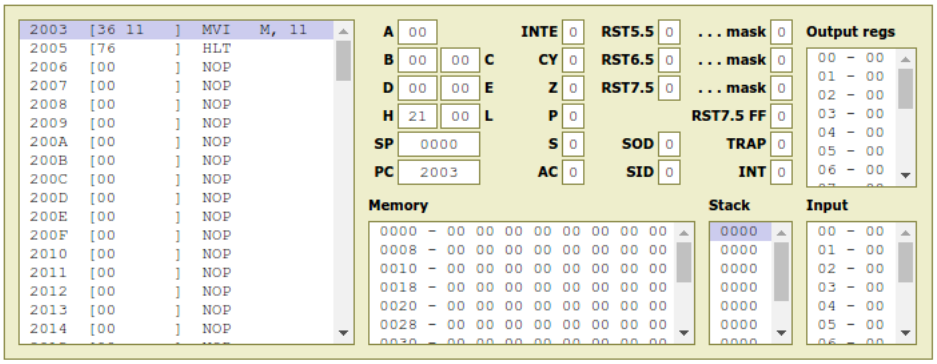



Ha be kell állítani a regiszterek kezdőértéket, akkor a megfelelő regiszterek mezőjébe írjuk bele a szükséges értéket.

### 3.1.3 3. lépés: kód követése

A bal oldali ablakban látható, hogy a PC a 2000h-ra mutat, ezt a kék háttérrel jelöli a rendszer (soron következő utasítás).

Ha rámegyünk a Step gombra az oldal tetején, akkor lefuttatja a PC által jelölt utasítást és az oldal alján elhelyezkedő táblázatban megjelenik néhány adat:




| Address | Data | $\overline{RD}$ | $\overline{WR}$ | IO/$\overline{M}$ | A | B | C | D | E | H | L | F [SZ-A-P-C] | SP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2000 | 21 | 0 | 1 | 0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00000000 | 0000 |
| 2001 | 00 | 0 | 1 | 0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00000000 | 0000 |
| 2002 | 21 | 0 | 1 | 0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00000000 | 0000 |

Vegyük sorra, hogy mi minden történt:

1. A bal oldali fehér ablakban a kék háttér most már a soron következő utasításra mutat (MVI M, 11), hiszen az előző utasítást már lefuttattuk és tovább lépett.

2. A táblázatban megjelentek az LXI H, 2100h utasítás során történt dolgok.

   (a) Mivel a 2000h-ra helyeztük el a kódot, a PC onnan olvassa fel az első adatot. Ott egy LXI utasítás van, ezért az első adat, amit felolvas az az utasítás opkódja.

   (b) Ökölszabály: minden utasítás első bájtja az opkódja.

   (c) Opkód: utasítás egyedi azonosító kódja

(d) Az LXI utasításhoz 2 dolog tartozik még: a regiszterpár, ahova adatot mozgatunk és maga az adat, amit mozgatunk. A regiszterpár az opkódba van kódolva, ezt a segédletből ki lehet olvasni. A másik paraméter, hogy milyen adatot akarunk elhelyezni a regiszterpárba. 2 bájtos adatot tudunk egy regiszterpárba tenni, ezért ez a 2 bájt az opkódot követő következő 2 címen helyezkedik el.

(e) Little-endian: kisebb címen kisebb helyiérték

(f) a 2100h számot szeretnénk a HL regiszterekbe tenni, szóval először felolvassuk a 2100h alsó bájtját (00h) a 2001h címről, majd a felső bájtját (21h) a 2002h címről.

(g) Egy utasítás opkódja és a paraméterei címfolytonosan helyekednek el a memóriában

(h) Címfolytonos: egymást követő memóriacímeken.

(i) Az opkód és az utasítás paramétereinek beolvasása mind olvasás művelet.

3. A Step-re kattintva lefuttatja a jelenlegi utasítást, ami az MVI M, 11h, majd megjelennek a táblázatban az eközben történtek:

   (a) Címfolytosan helyezkedik el a kód, szóval a következő címen van az MVI M utasítás opkódja, vagyis a 2003h-n.

   (b) Az MVI M utasításhoz tartozik egy paraméter: milyen adatot akarunk elhelyezni az M által mutatott memóriacímre. Ezt az adatot az opkódot követő címről olvassa fel, tehát a 2004h-ról.

   (c) Felolvastunk az MVI M, 11h-hoz tartozó minden adatot, szóval el tudjuk ténylegesen végezni az utasítást fizikailag is: elhelyezzük az M által mutatott memóriacímre a 11h-t.

   (d) Az M pointer a HL regiszterpár tartalmával jelölt memóriacímre mutat, a HL-ben jelenleg a 2100h van, hiszen az imént tettük bele. Így a 2100h címre írjuk a 11h-t. Ez látszik a táblázatban is.

   (e) Figyeljük meg, hogy a $\overline{WR} = 0$, mert ez egy adat írása egy adott memóriacímre. Az Address oszlopban a 2100h szerepel, hiszen erre a címre írunk adatot. A Data oszlopban a 11h érték szerepel, mert ezt az adatot írjuk oda.

   (f) Látható, hogy a H és L oszlopokban a regiszterpár tartalma az előbb odahelyezett 21h és 00h.

4. Ismételten a Step-re nyomva lefut a HLT utasítás is. A HLT címfolytosan helyezkedik el (hiszen nem volt például ORG direktíva vagy hasonló), tehát a 2005h-n. Az adat csupán az utasítás opkódja, mert ehhez az utasításhoz nem tartozik semmilyen paraméter sem.

5. Végeztünk, elvileg most megtanultuk használni a szimulátort. Ezek után sok gyakorlással ezekkel az alapokkal már ügyesen fogunk tudni bánni ezzel a hasznos kis segédeszközzel.