

1.MÉRÉS

8051 mikrokontroller programjának ellenőrzése szimulátorral

1. A mérés célja:

A technológiai folyamatok irányítására is alkalmas mikrokontroller (PBC80552)- bázisú mikrogép programozásának és tesztelésének megismerése. Az Assembly-nyelvű programozás alapjainak gyakorlása.

2. A szükséges ismeretek:

Szimulátor program a 8051/31 mikrokontroller családhoz (SIMULA5X)

A PC-n futó szimulátorral a fejlesztett programot ellenőrzött módon lehet futtatni. Ez azt jelenti, hogy a program végrehajtása közben:

- meg tudjuk vizsgálni és esetleg módosítani a kontroller bármelyik regiszterének tartalmát (watch),
- meg tudjuk a program futását adott helyen állítani, azaz töréspontot tudunk elhelyezni (breakpoint),
- tudjuk a programot lépésenként futtatni (single step),
- a külső környezetből jövő (input) adatokat fájlból beolvasva szimulálni (stimulus),
- illetve kiküldött (output) adatokat fájlba eltárolni,
- a szimuláció során bekövetkező eseményeket a későbbi elemzés céljából gyűjteni (log).

Mivel a program támogatja az egér használatát, ezért a szokásos módon egérrel és billentyűzettel is kezelhető.

A szimulációs programban lehetőség van az assembly nyelvű forrásnyelvi program írására, és a megírt program futtatására, szimulálására. Ennek segítségével rövidebb programok, szubrutinok megírása és ellenőrzése is elvégezhető a szimulációs környezetben.

A szimulációs programcsomagban lévő disassembler segítségével módunk van közvetlen gépi kódú programok (EPROM tartalom) visszafejtésére, program elemzésére.

Intel Hex, illetve bináris formátumú programok beolvasását, illetve fájlba írását is támogatja a program.

A program elindítása után a következő képet látjuk a számítógép képernyőjén:

Gépi kódú program:
cím_utasításkód_operandus

A legfontosabb regiszterek neve, alattuk az aktuális tartalmuk

Forrásnyelvi lista

File Run View Break Assemble Options F1=Help

R0 R1 R2 R3 R4 R5 R6 R7 A B DPTR PC PSW	Flags
00 00 00 00 00 00 00 00 00 00 0000 0000 00	- - - RB0- -

PROG:

0000: 02 00 0C LJMP INIT1

0003: 75 D0 00 MOV PSW, #00

INTRCL:

0006: 78 FF MOV R0, #FF

VICL:

0008: 76 00 MOV @R0, #00

000A: D8 FC DJNZ R0, VICL

INIT1:

000C: 75 81 60 MOV SP, #60

000F: 90 5F FF MOV DPTR, #5FFF

0012: C2 03 CLR GYU

0014: 12 01 82 LCALL LACINIT

MAIN:

0017: 12 01 95 LCALL PICOLV

001A: F5 F0 MOV B, A

SP= 07 07 07

FF: 00 00 00 00

03: 00 00 00 00

SP: 00 00 00 00

OP-2:0000

OP-1:0000

P0 = 11111111

P1 = 11111111

P2 = 11111111

P3 = 11111111

P4 = 11111111

PWM0:0 PWM1:0 W

-IData -0--1--2--3--4--5--6--7---8--9--A--B--C--D--E--F-IData-SFR-XData-Code-

0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *.....*

0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *.....*

0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *.....*

F2=SFR, F4=Goto Addr, F7=Step, F8=Proc, F9=Run, Alt F., F10=Menu 80552/83552

A verem memória tartalma

Az előzőleg végrehajtott utolsó két utasítás címe

Portok tartalma

Memóriatartalom kijelzése:
Idata - belső RAM
SFR-SFR regiszterek
Xdata - külső RAM

Az egyes menük megnyitása vagy **egérrel** rákattintással, vagy az **ALT** és a menü **nagybetűvel** írt betűjének együttes lenyomásával történhet pl. a File menü **ALT+F** betűkombinációval nyílik.

A következőkben az egyes menükön belüli alpontok jelentését tekintjük át.

File menü parancsai:

Load Intel Hex file	Intel hexa formátumú fájl betöltése. (ld. később!)
Load binary file	Bináris fájl betöltése - az a fájl a kódmemória bináris alakját tartalmazza
Load symbol file	Szimbólum fájl betöltése - a forrásprogramban használt címkék, változók nevei
Clear symbol stack	A szimbólumokat tartalmazó verem törlése
Write Intel Hex file	A memória tartalmának Intel hexa formátumú fájlba írása
Write binary file	A memória bináris tartalmának fájlba írása
Load stimulus file	A bemeneti gerjesztéseket tartalmazó fájl betöltése
Open log file	A szimuláció alatt történt eseményeket naplózó fájl megnyitása
DOS command	DOS parancs kiadása a programból való kilépés nélkül
Quit ALT-F4	Kilépés a programból

Run menü parancsai:

Single step F7	A program lépésenkénti végrehajtása
Single proc F8	A program lépésenkénti végrehajtása, szubrutinok is egy lépésben
Run program F9	A program futtatása
Goto address F4	A program adott címére ugrás
Restart user prog	A újra indítása

A **lépésenkénti program végrehajtás**nál a felhasználói programnak mindig csak egy utasítása hajtódik végre, és ilyenkor mindig lehetőség van a regiszterek vagy a tártartalmak vizsgálatára, vagyis a végrehajtott utasítás hatásának, a program helyes működésének az ellenőrzésére.

Ha a töréspontok elhelyezésével sikerült meghatározni azt a programrészt, ahol a program hibásan működik, akkor ezen a részen belül a hibás utasítás vagy utasítássorozat megtalálása a lépésenkénti program végrehajtás segítségével könnyen elvégezhető. Lehetőség van szubrutinok egy lépésben történő végrehajtására is a **Single proc** paranccsal.

View menü parancsai:

Register, R0 ...	R0-R7 regisztertömb kijelzése/módosítása
Ports, P0 ...	Portok kijelzése/módosítása

Internal RAM, Idata

SFR	Speciális funkciójú regiszterek
Extern. memory, Xdata	Külső adatmemória megnézése/módosítása
Code memory, Cdata	Program memória megnézése/módosítása

Idata symbols	Belső RAM - hoz a programban rendelt szimbólumok megnézése
SFR symbols	SFR - ekhez a programban rendelt szimbólumok megnézése
Bit symbols	Bitekhez a programban rendelt szimbólumok megnézése
Code symbols	A programban használt szimbólumok megnézése
Xdata symbols	Külső RAM - hoz a programban rendelt szimbólumok megnézése

Serial input	Soros bemenetre küldött adatok
Serial output	Soros kimeneten megjelenő adatok
Cycle count	Ciklusszám megjelenítése

Break menü parancsai:

Set breakpoint ctrl - B	Töréspont beállítása
Clear breakpoint ctrl - E	Töréspont törlése
Remove all breakpoints	Az összes beállított töréspont eltávolítása
List breakpoints	Töréspontok listázása

A felhasználónak a program élesztése során igen nagy segítséget nyújt, ha a programot egy, még az indítás előtt kijelölt címen meg lehet állítani, azaz a futását felfüggeszteni. Ekkor ugyanis a regiszterek és a programban használt változók tartalmainak vizsgálatával könnyen eldönthető, hogy a program eddig a pontig helyesen, vagy hibásan működött. A teljes programot több, logikailag jól elhatárolt részre bontva, a hibásan működő programrész gyorsan meghatározható. Az ilyen leállási feltétel - azaz töréspont (angolul breakpoint) - a programban több is elhelyezhető.

Assemble menü parancsai:

Assemble	Assembler forráskód beírása
Disassemble screen	Kódból visszafordított assembler forrás képernyőre írása
Disassemble PRN	Kódból visszafordított assemblerforrás nyomtatóra
Move code memory	Program memória átmozgatása
Fill code memory	Program memória feltöltése

A fejlesztői munka során gyakori, hogy egy gépi kódú programrészlet működését kell megérteni. A gépi kódból az utasítások visszafejtése, egy aránylag egyszerűen programozható tevékenység. Lényegében a fordítóprogram (az assembler) működésénél leírtak "visszafelé történő" végrehajtására, sőt ennél egy kicsit egyszerűbb műveletre kell gondolni. Az assembler ugyanis az állandó és a változó szimbólumtáblát is kezeli, vagyis a felhasználó által definiált szimbólumok értékeit nyilván kell tartania és a fordítás során a keletkező gépi kódba ezt be kell szerkesztenie.

A visszafordítás (a disassemblálás) során csak az állandó szimbólumtáblát kezeljük. Az operandus mezőben szereplő értékeket nem szimbolikusan, hanem abszolút formában, hexadecimális számként jelezzük ki. A visszafordító program működése tehát viszonylag egyszerű. Az első (vagy első két) bájtból megállapítja, hogy milyen utasításról van szó, a táblázat alapján kiírja a mnemonikáját, értelmezi és visszafordítja az operandus mezőben szereplő értéket.

A probléma azonban ott kezdődik, ha nem tudjuk, hogy melyik az utasítás első bájtja. Ugyanez az eset áll elő akkor is, ha ismerjük ugyan a program kezdetét, de a program utasításai között adatterület is elhelyezkedik. Mindkét eset azt eredményezi, hogy a visszafordító program adatot értelmez utasításként, és így hibásan fordít vissza. Ezért minden visszafordítást kritikával kell fogadnunk. Ellenőriznünk kell, hogy nem tévedt-e el a visszafejtő program. Például gyakran előfordul, hogy egy programrészben egymás után sok **MOV** utasítás szerepel. Ez a terület szinte biztosan nem programot tartalmaz, hanem szöveget, mivel az **MOV** utasítás kódjai éppen az **ASCII** kódtáblában értelmezett kódok tartományába esnek. Ennek gyors ellenőrzésére a legtöbb visszafejtő program egyszerű lehetőséget biztosít azzal, hogy a visszafejtett listán a bájtoknak megfelelő **ASCII** kódot is megjeleníti, a nem ábrázolható kódokat egy "."-al jelezve.

A tapasztalat azt mutatja, hogy a visszafordítás az adatterület vége után 5-6 bájtal helyreáll. Ez annak köszönhető, hogy az **MCS-51** utasításai között aránylag sok az egybájtos utasítás. Egy ilyen területre "ráfutva" a visszafordítás ismét helyes lesz. Mivel a maximális utasításhossz négy bájt lehet, ezért elegendő három egymást követő egybájtos utasítás ahhoz, hogy a "tévelygés" megszűnjön. Igényesebb visszafejtők, ha előre megadjuk, akkor a táblázatos részt nem utasításként, hanem táblázatként adják vissza a listában. Másik fejlett szolgáltatásuk abban áll, hogy a visszafejtett listát--ami tulajdonképpen egy szövegfájl--képes a rendszer szövegszerkesztője fogadni, és ebből a visszafejtett listából fordítható forrásnyelvi listát készíteni.

Options menü parancsai:

Szimulálandó processzortípus kiválasztása

MCU 8051/8031
MCU 8052/8032
MCU 83C552/80C552
MCU 80515/80535
MCU 80C517/80C537

Reset mikrocontroller	Kontroller RESET - elése
80C552 Watchdog en.	80c552 Watchdog áramkörének engedélyezése
80C517 data pointer	80c517 adatmutatójának beállítása (mivel több van)
80C517 PE#/SWDT-pin	láb beállítása
A/D - Converter voltage	A/D átalakító feszültsége - itt adhatók meg a bemeneti feszültségek értékei.
Ports for log file	Portok naplózása
Serial I/O ASCII	a soros vonal bájttjainak ASCII alakja

Az Intel hexa formátum

Az Intel hexa formátumnál a program gépi kódú részét rekordokba szervezik. Egy rekord változó hosszúságú lehet.

A rekordban az első mező a **rekordjelző mező (Record Mark Field)**. Ez a mező jelzi a rekord kezdetét és egy ASCII kettőspontot tartalmaz (:).

A második mező a **rekordhossz mező (Record Length Field)**. Ez a mező két ASCII karaktert tartalmaz, melyek jelzik a rekordban lévő adatbájtok számát. A hexadecimálisan adott adatbájtok számát két ASCII karakterre konvertálva adódik ki a mező két karaktere, a magasabb helyiértékű digit szerepel előbb. Egy rekordban maximálisan 255 adatbájt lehet.

A harmadik mező a **betöltési cím mező (Load Address Field)**. Ez a mező négy ASCII karaktert tartalmaz, a rekord hexadecimálisan adott betöltési címének ASCII karakterre konvertált értékét az alábbi sorrendben: a cím felső bájttjának magasabb helyiértékű digitje, a cím felső bájttjának alacsonyabb helyiértékű digitje, a cím alsó bájttjának magasabb helyiértékű digitje, a cím alsó bájttjának alacsonyabb helyiértékű digitje. A rekordban lévő első adatbájt a betöltési címre töltődik, az utána következő adatbájtok a sorban következő címekre. A fájlvége (End Of File, EOF) rekordban ez a mező négy ASCII nullát tartalmaz vagy a program kezdőcímét.

A negyedik mező a **rekord típus mező (Record Type Field)**. A rekord típus adatrekord esetén 00, fájlvége rekord esetén 01. Ez a mező két ASCII karaktert tartalmaz, a rekord típus ASCII karaktereit, a magasabb helyiértékű digit szerepel előbb.

Az ötödik mező az **adat mező (Data Field)**. Ez a mező tartalmazza az aktuális adatokat két-két ASCII karakterre konvertálva, a magasabb helyiértékű digit szerepel előbb. A fájlvége rekordban nincs adat mező.

Az utolsó mező az **ellenőrző összeg mező (Checksum Field)**. Az ellenőrző összeg a második, harmadik, negyedik és az ötödik mező hexadecimális bájttjainak 8 bitre csonkított összegének kettes komplementens - e. Az így kapott összeget két ASCII karakterre konvertálva kapjuk a mező két karakterét, a magasabb helyiértékű digit szerepel előbb.

A leírásból is látható, hogy egy Intel hexa formátumú fájl egy szöveges (ASCII) fájl, így pl. egy szövegszerkesztő vagy listázó program segítségével megvizsgálhatjuk a tartalmát. Példa Intel hexa formátumra:

:10200000455A5420455244454D455320564F4C5453

:0B201000204D454746454A54454E49C7

:00000001FF

A leíráshoz kapcsolódva, az első sor mezői:

1.	2.	3.	4.	5.	6.
:	10	2000	00	455A5420455244454D455320564F4C54	53

3. Házi feladatok:

3.1. Írja meg az alábbi **adatmozgató** feladatok programjainak forrásnyelvű változatát:

- töltse fel az akkumulátort **82** decimális értékkel, majd vigye az **akkumulátor** tartalmát az **55H** című bájtbá **direkt**, illetve az **56H** című memóriarekeszbe **indirekt** címezéssel,
- **ciklusba** szervezéssel töltse fel a **20H** címen kezdődő **60** bájtot **0FFH** értékkel,
- cserélje meg az **A** és **B** regiszterek tartalmát **veremkezelő** (Stack) utasítások felhasználásával,
- másolja át a **külső adatmemória 4000H** címtől kezdődő **5** bájtnak a tartalmát a belső adatmemóriába a **60H** címen kezdődő területre,
- a **kódmemória** első **5** bájtnak másolja a belső adatmemória **70H** címmel kezdődő területére.

3.2. Írja meg egy olyan program forrásnyelvű fájl-ját, amely képezi a **P1** , illetve a **P4** portokon bevitt két szám:

- összegét, különbségét, szorzatát, hányadosát és maradékát, és
- ezeket sorrendben lehelyezi a **40H** címtől kezdődően .

4. Mérési feladatok:

4.1. Hozzon létre az **XE251\MUNKA** könyvtárban belül egy olyan alkönyvtárt, melyet továbbra is a mérőcsoport használ (pl. *PSCI_I*).

4.2. Indítsa el a **Simula5x** programot, és írja be a házi feladatban elkészített programokat a **4000H** címtől kezdve egymás után. Az egyes programrészeket két-két NOP utasítással válassza el.

4.3. Lépésenkénti üzemmódban ellenőrizze az egyes feladatokat megvalósító programrészek helyes működését.

4.4. A működést ellenőrizze töréspontos futtatással is,

4.5. A hibátlan programokat Intel hexa formátumban vegye fel a megnyitott alkönyvtárba.

5. Kérdések

5.1. Milyen címezési módokat ismer a mikrokontrollernél?

5.2. Hogyan kezeli a mikrokontroller a külső program-, és adatmemóriát?

5.3. Hogyan valósítható meg program futtatása RAM-ból?

5.4. Mit tartalmaz egy program lista fájlja?

5.5. Mit jelent a lépésenkénti és a töréspontos program ellenőrzés?