1.Suport teoretic

Cuprins

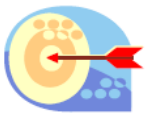
Cuprins Laborator 1

Cunoştinţe iniţiale

Pentru a putea rezolva problemele puse la această serie de lucrări de laborator este nevoie ca studenții să recapituleze noțiunile electronică digitală, de arhitectură a sistemelor de calcul şi de programare în limbaj de asamblare.

Introducere

Prima parte a şedinței de laborator este dedicată verificării cunoştinţelor studenților prin discuții. Discuțiile punctează principalele noțiuni care trebuie recapitulate.
A doua parte a şedinței prezintă ideea directoare a laboratorului, modul de lucru la laborator, modul de notare şi modul de examinare.



Obiective

După parcurgerea acestui modul studenții vor avea o imagine:

- A noțiunilor pe care trebuie să le repete pentru a putea parcuge cu succes lucrările următoare;
- A desfăşurării laboratorului, a softului folosit şi a elementelor de programare utilizate;
- A modului de apreciere şi de examinare.

Durata medie de studiu individual

Durata medie de studiu individual este de 2 ore.

1.1.Descrierea setului de lucrări

Pentru aceste lucrări de laborator a fost conceput un dispozitiv care se poate cupla la un calculator PC prin interfața paralelă sau USB. Dispozitivul conține un port de ieşire echipat cu 8 diode LED şi un port de intrare echipat cu 4 microîntrerupătoare. Cu un software simplu se pot da comenzi de aprindere a LED-urilor şi se poate citi starea microîntrerupătoarelor. Programarea secvențelor de intrare şi ieşire de date se face în limbaj de asamblare, acest limbaj fiind cel mai apropiat de structura hardware a calculatorului. O fotografie a dispozitivului cuplat la un calculator este dată în figura 1.1:

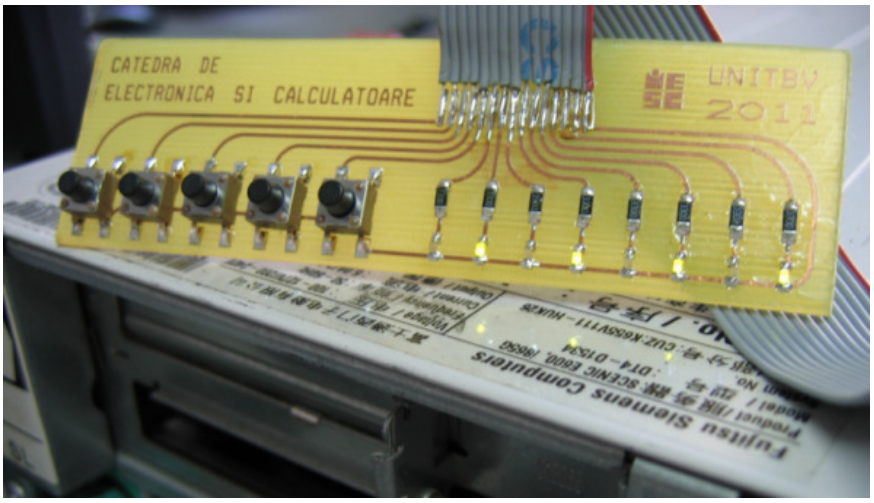


Figura 1.1. Dispozitivul cu LED-uri folosit la laborator

În prima lucrare se revăd noțiunile teoretice legate de conversiile numerelor exprimate în baza 10, 2 şi 16 precum şi funcționarea unei unități centrale, focalizând explicațiile pe registre, porturi de intrare ieşire şi instrucțiuni în limbaj de asamblare pentru microprocesoarele din familia x86. Este prezentat de asemenea un program cu care se pot rula programe simple şi scurte. Avantajul utilizării programului utilizat este faptul că studenții pot observa la rularea pas cu pas conținutul fiecărui registru, pot observa schimbarea registrului IP (Instruction Pointer) în cazul unui salt şi pot vedea dacă programul se comportă conform aşteptărilor.

În următoarele şedințe de laborator se rulează programe care aprind LED-urile în anumite secvențe întâi pas cu pas, apoi la viteza procesorului. Programele devin în fiecare şedință mai complexe şi mai interesante, fără însă a depăşi 20, 30 de instrucțiuni. Programele realizate aprind LED-uri în diferite secvențe şi citesc întrerupătoare. După ce studenții pot să lucreze cu ambele porturi ale dispozitivului cu LED-uri urmează câteva lucrări de studiu al conectării şi programării unor circuite pe magistrală.

Ultimele lucrări trec la un alt nivel şi studenții abordează programarea interfețelor de comunicații prin comenzi AT.

Pe parcursul şedințelor de laborator se acordă bonificații pentru cele mai bune şi mai rapide realizări. Laboratorul se termină cu un test de laborator.

1.2.Recapitularea noțiunilor importante

În **arhitectura calculatoarelor**, un **registru de procesor** este o cantitate mică de spațiu de stocare disponibilă în unitatea centrală de procesare, spațiu al cărui conținut poate fi accesat mai rapid decât datele aflate în altă parte (de exemplu, în **memoria principală**). Registrul este caracterizat de numărul de biți: 8, 16 sau 32 de biți. Registrele microprocesoarelor din familia x86 (INTEL) sunt:

- AX, BX, CX, DX de 16 biți care pot fi împărțite în câte două registre pe 8 biți,
- AH, AL, BH, BL, CH, CL, DH, DL şi registrele pe 16 biți: SI, DI

Porturile sunt registre de acces în exteriorul microprocesorului, caracterizate de numărul de biți.

Limbajul de asamblare este limbajul cel mai apropiat de structura arhitecturală a unui microprocesor, dar şi limbajul cel mai "primitiv" şi îndepărtat de modul de comunicație uman.

De exemplu instrucțiunea:

MOV AL, BL mută conținutul registrului de 8 biți BL în AL (întotdeauna operandul din stânga este destinația iar cel din dreapta sursa)

Unele registre pot avea semnificații dedicate:

- registrul DX se foloseşte pentru a indica adresa unui port (Address Pointer) iar
- registrul BX o adresă de memorie (Memory Address)

Presupunem că la portul cu adresa 0378H se află conectate 8 LED-uri. Secvența de program următoare are ca efect stingerea LED-urilor.

```
MOV DX, 0378H    ; valoarea hexa 0378 se încarcă în registrul DX
MOV AL, 00H      ; valoarea 00H se încarcă în registrul AL
OUT DX, AL       ; valoarea 00 este trimisă la portul cu adresa 0378H
```

Secvența următoare are ca efect aprinderea celor 8 LED-uri:

```
MOV DX, 0378H    ; valoarea hexa 0378 se încarcă în registrul DX
MOV AL, FFH      ; valoarea FFH se încarcă în registrul AL
OUT DX, AL       ; valoarea FF este trimisă la portul cu adresa 0378H
```

Adunare şi scădere

ADD AL, BL ; AL <= AL + BL
SUB AL, BL ; AL <= BL - AL

! Registrul care stochează întotdeauna rezultatele se numeşte Acumulator

Înmulțire şi împărțire

MUL AL,BL ; AL <= AL * BL
DIV AL,BL ; AL <= AL / BL

Operații logice

AND AL, BL ; AL <= AL & BL
OR AL, BL ; AL <= AL | BL
XOR AL, BL ; AL <= AL ^ BL

Stocarea în memorie

MOV BX, 1000H ; BX <= 1000H
MOV [BX], AL ; se salvează conținutul AL la adresa de memorie conținută în BX (mem[BX] <= AL)

Condițiile se referă la rezultatul operației aritmetice anterioare şi în funcție de acestea se pot face salturi:

JP ; salt dacă "pozitiv"
JNZ ; salt dacă "diferit de zero"
JE ; salt dacă "egal"
JNE ; salt dacă "diferit"

Setarea (stabilirea) unei condiții se poate face cu o instrucțiune de comparare:

CMP AL, BL ; compară conținutul registrelor AL şi BL prin scădere.

Dacă sunt egale se setează condiția de **zero** şi **egal**.

1.3. Programul Symbolic Instruction Debugger (SID)

Programul SID este un program cu ajutorul căruia se poate foarte simplu rula şi depana un program în limbaj de asamblare. Comenzile programului pot fi afişate cu comanda ? după intrarea în program şi afişarea prompteruluzi SID (#):

```
#?
?      Help
??     Command formats
:name  Define a macro        (You're prompted to enter the macro body)
=name  Invoke a macro        =      List all defined macros.
A      Assemble into memory
B      Block compare
D      Display memory        -D     Set default nr of bytes to display
E      Load (for Execution) program & symbol file[s]
```

```
F       Fill memory
G       Go (with optional temporary breakpoints)
H       Hexadecimal arithmetic
I       Set up program arguments
L       List memory (disassemble)
M       Move (copy) memory block
P       Pass points (i.e., breakpoints)
Q       Direct I/O request - Or QUIT
R       Read disk file
S       Set memory              SR      Search
T       Trace (single-step execution)
U       Untraced single-step execution
V       Verify values of last file loaded
W       Write disk file
X       Examine [or modify] CPU state
Z       Display 8087 Math Co-processor registers
```

La comanda ?? se afişează formatul comenzilor SID:

```
#??
In the formats below, `s' means a full (base and offset) address, `f' is
an offset, and `n' is a number.  Brackets surround optional elements.
A s             Assemble into memory, at address s
B s1,f,s2       Block compare memory at s1 (through f) vs block at s2
D[W]s[,f]       Display memory (-D sets default nr of bytes to display)
E file1 [[-]file2]      Load files and symbols; E alone frees memory
      try `-' or `+' preceding file2 for `large' symbol addresses
F[W]s,f,n       Fill memory from s to f with n
[-]G[s1][,s2[,s2]]      Go at s1; s2 & s3 are breakpoints; `-' quiets it.
H[n1[,n2]]      Show symbols; show n1; show n1+n2, n1-n2, n1*n2 and n1/n2
Istring         Set up command tail (command line arguments)
[-]L[s[,f]]     List memory (disassemble) from s to f; `-' won't show symbols
Ms1,f,s2        Move (copy) memory block from s1 (through f) to s2
[-]P[s[,n]]     Set/clear/list breakpoints (called "pass points" here)
QI[W]n1 or QO[W]n1,n2       Input from (or output n2 to) port n1 ...or...
QRs,drive,sector,count or QW... Read/Write count sectors from drive to s
Rfile           Read disk file
[-]S[W][s]      Store into memory or enable/disable segment register display
SRs,f,"string"  Search through memory
[-]T[W][n]      Trace/step execution
[-]U[W][n]      Untraced step execution
V               Values of last file read
Wfile[,s,f]     Write memory (block from s to f) to disk file
X[R|S]          Examine CPU state [alter:  R == registers; S == a state flag]
Z       Display 8087 Math Co-processor registers
```

Comenzile SID se pot scrie după prompterul SID care este **#**

Cele mai folosite instrucţiuni sunt:

Scrierea unui program începe cu: **A<adresa>** (scriere în asamblare a unei instrucţiuni de procesor la **adresa** specificată. Zona de adrese posibile sunt de la 0000H la FFFFH). După scrierea ultimei instrucţiuni se iese în prompterul SID cu . (punct) şi ENTER.

Rularea instrucțiune cu instrucțiune se face prin stabilirea adresei de start cu **XIP** ENTER, urmat de adresa instrucțiunii de executat. Apoi **T** urmat de ENTER. Un **T** în continuare înseamnă execuția următoarei instrucțiuni.

Rularea continuă se comandă cu **G<adresa>**.

Salvarea programului se face cu comanda **W<numeFisier>, <adresaÎnceput>, <adresaSfârşit>**

Încărcarea unui fişier salvat se face cu comanda **R<numeFişier>**

Cu **?** se afişează comenzile SID şi cu **??** se afişează amănunte.

Lansarea în execuție a unui program care rulează în buclă produce blocarea aplicației SID şi trebuie închisă fereastra.

Cu **X** se poate vizualiza conținutul registrelor microprocesorului.

Cu **L<adresa>** se poate vizualiza programul scris, câteva instrucțiuni. Încă un L şi ENTER afişează următoarele instrucțiuni.

A nu se confunda comenzile SID (T, A, X) cu instrucțiunile microprocesorului (MOV, OUT).

1.4.Deblocare porturi sub XP

Sistemul de operare Windows XP protejează accesul direct la porturi din rațiuni de securitate. La laborator se foloseşte accesul direct, de aceea porturile trebuie deblocate. La laborator programele de deblocare şi documentația se vor afla pe desktop (folderul **UserPort).** Din acest folder (subdirector) se copiază fişierul **UserPort.SYS** în subdirectorul **WINDOWS\SYSTEM32\DRIVERS.** Se lansează UserPort şi apare fereastra din figura 1.2:

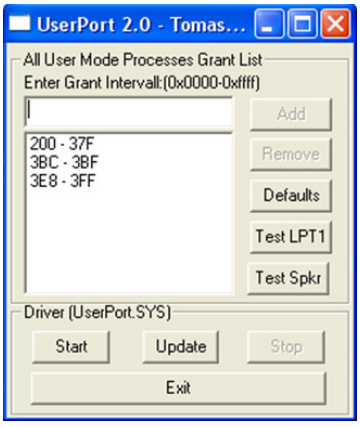


Figura 1.2.Fereastra de deblocare selectivă a porturilor

Se alege zona de adresare care trebuie eliberată, 0200-037F şi se apasă Add, apoi Start. Aplicația se poate închide. După eliberarea porturilor se poate lansa aplicația SID.

1.5. Placa cu LED-uri pentru laborator

Placa cu LED-uri folosită la laborator are schema bloc din figura 1.3.

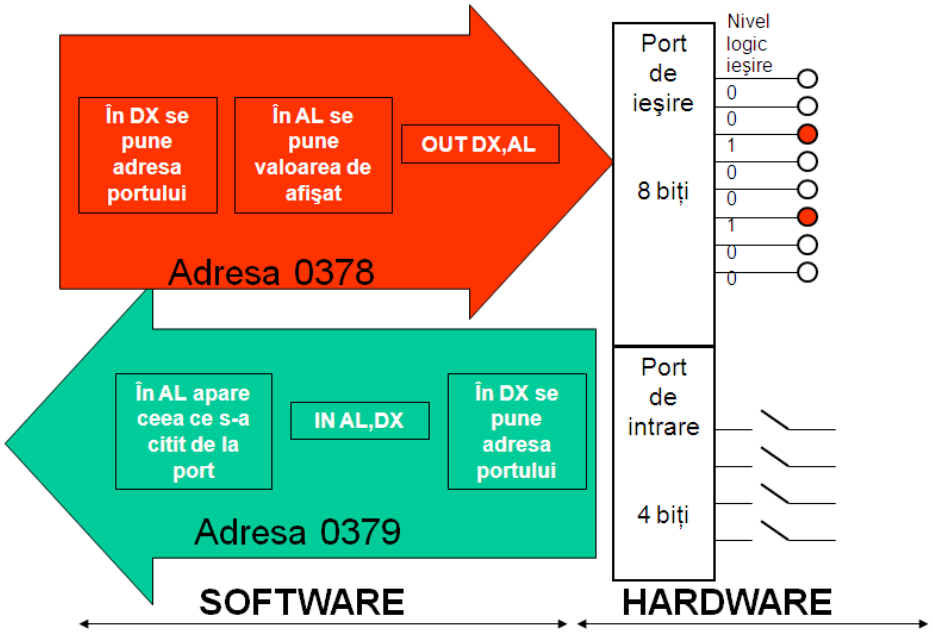


Figura 1.3. Schema bloc a plăcii cu LED-uri

Placa conține două porturi, unul de ieşire la adresa 0378H la care sunt suplate 8 LED-uri şi un port de intrare la adresa 0379H la care sunt conectate 4 întrerupătoare. Succesiunea operațiilor pentru citirea şi scrierea în port sunt date pentru fiecare port în figură.

1.6.Tema laboratorului, desfăşurarea şi testul

Scopul laboratorului este ca studenții să realizeze un program pentru aprinderea LED-urilor într-o succesiune cât mai plăcută ochiului, la dorința şi conform imaginației

fiecăruia. La apăsarea unui buton succesiunea de aprindere să se modifice. După revenirea butonului succesiunea poate reveni la cea inițială sau poate rămâne cea modificată, la dorința fiecăruia. Cei care se consideră foarte buni, pot implementa două modificări, care apar la apăsarea a 2 butoane diferite, dar să fie atenți să nu se complice atât de mult încât să nu mai meargă nimic. Nota va ține cont de viteză şi de originalitatea şi frumusețea secvenței de aprindere.

La fiecare laborator un student va primi o bonificație, cel care termină primul sarcina, cel care realizează secvența cea mai spectaculoasă sau cel care are un răspuns la o întrebare dificilă. Uneori se pot acorda mai multe bonificații la o şedință de laborator.

Laboratorul se termină cu un test de laborator la care studenții trag un bilet cu tema pe care trebuie să o rezolve. Biletul solicită realizarea unui program care conține realizarea a 3 secvențe dinamice şi trecerea între ele prin apăsarea a 2 switch-uri. Nota este acordată în funcție de stadiul programului la sfârşitul şedinței de laborator astfel: doar secvențe dinamice - 5, comutarea la apăsarea unui microîntrerupător - 7, comutarea la apăsarea ambelor microîntrerupătoare - 10. Timpul alocat realizării testului este de 100 de minute.

Câteva sfaturi utile:

1. Salvați programul cu comanda **SID W** înainte de a-l lansa în execuție. La salvare fiți atenți să salvați tot programul, deci atenție la adresa de început şi sfârşit.
2. Ca să verificați că programul funcționează, rulați prima dată programul pas cu pas şi doar pe urmă dați **G**.
3. Încercați prima dată programe simple, apoi treceți treptat la variante mai complexe.
4. Când **SID** încarcă un program salvat îl pune la adresa **0000H** şi modifică toate adresele de salt ca să fie corecte.
5. Atenție la adresele puse de **SID** la fiecare instrucțiune pentru că dacă scrieți un salt trebuie să ştiți adresa de salt!
6. Dacă după scrierea unei instrucțiuni **SID** răspunde cu **?** atunci sintaxa instrucțiunii este eronată.
7. Înainte de a rula programul cu **L<adresa>** verificați programul scris.

Tema abordată la laborator permite şi încurajează lucrul în colaborare. Studenții care înțeleg mai repede pot explica celor care nu au înțeles. Viteza cu care înțeleg studenții este diferită şi depinde de cunoştințele lor de software anterioare şi de pasiunea lor pentru programare. Studenții care se descurcă mai greu la programare vor înainta mai încet dar vor putea ajunge la rezultatele dorite. Este important ca aceştia să respecte etapele laboratorului, să înțeleagă şi să realizeze sarcinile din fiecare etapă.

Rezumat

În prima lucrare se recapitulează noțiunile teoretice legate de conversiile numerelor exprimate în baza 10, 2 şi 16 precum şi funcționarea unei unități centrale, focalizând explicațiile pe registre, porturi de intrare ieşire şi instrucțiuni în limbaj de asamblare pentru microprocesoarele din familia x86. Este prezentat de asemenea un program *debugger* cu care se pot rula programe scurte şi se poate vedea efectul execuției fiecărei lii de program executate. Se dau câteva sfaturi utile pentru a putea utiliza cu uşurință acest program

Este prezentat dispozitivul cu LED-uri şi întrerupătoare care va fi folosit la laborator şi se descrie modul de desfăşurare a laboratorului.

Bibliografie

1. Petre Ogrutan, Carmen Gerigan, *Memorii, interfete si periferice*, Indrumar de laborator, Reprografia Universitatii 1998, online la: http://vega.unitbv.ro/~ogrutan/lab/index.html
2. P. A. Carter, *PC AssemblyLanguage, 2003* http://pdos.csail.mit.edu/6.828/2012/readings/pcasm-book.pdf