8. Tétel

a. Szoftveres megszakításkezelés terminál input-output, az Assembly és a Pascal nyelv kapcsolata:

Mint említettük, szoftver-megszakításnak egy program által kiváltott megszakítást nevezünk. Hogy miért van erre szükség? Nos, néhány szoftver-megszakítás jócskán megkönnyíti a különféle hardvereszközök elérését, míg mások az operációs rendszer egyes funkcióival teremtenek kapcsolatot, de számos egyéb felhasználásuk is lehetséges.

Összesen 256 db. szoftver-megszakítást kezel a 8086-os mikroprocesszor. Ebből 8 (vagy 16) db. az IRQ-k kezelésére van fenntartva, néhány pedig speciális célt szolgál, de a többi mind szabadon rendelkezésünkre áll. Azt azért tudni kell, hogy a gép bekapcsolása, ill. az operációs rendszer felállása után néhány megszakítás már foglalt lesz, azokon keresztül használhatjuk a munkánkat segítő szolgáltatásokat.

Megszakítást az egyoperandusú INT utasítással kezdeményezhetünk. Az operandus kizárólag egy bájt méretű közvetlen adat lehet, ami a kért szoftver-megszakítást azonosítja. Az utasítás pontos működésével a következő fejezetben foglalkozunk majd. Érdemes megemlíteni még egy utasítást. Az INTO (INTerrupt on Overflow) egy 04h-s megszakítást kér, ha OF = 1, különben pedig működése megfelel egy NOP-nak (azaz nem csinál semmit sem, csak IP-t növeli). Ennek akkor vehetjük hasznát, ha mondjuk az előjeles számokkal végzett műveletek során keletkező túlcsordulás számunkra nemkívánatos, és annak bekövetkeztekor szeretnénk lekezelni a szituációt.

Egy adott számú megszakítás sokszor sok-sok szolgáltatás kapuját jelenti. A kívánt szolgáltatást és annak paramétereit ilyenkor különböző regiszterekben kell közölni, és a visszatérő értékeket is általában valamilyen regiszterben kapjuk meg. Hogy melyik megszakítás mely szolgáltatása hányas számú és az milyen regiszterekben várja az adatokat, lehetetlen fejben tartani. Ezeknek sok szakkönyvben is utánanézhetünk. Ha viszont a könyv nincs a közelben, akkor sem kell csüggedni. Számos ingyenes segédprogramot kifejezetten az ilyen gondok megoldására készítettek el. A legismertebbek: HelpPC, Tech Help!, Norton Guide, Ralf Brown's Interrupt List. A programok mindegyike hozzáférhető, és tudtunkkal magáncélra ingyenesen használható. Mindegyik tulajdonképpen egy hatalmas adatbázison alapul, amiben hardveres és szoftveres témákat is találhatunk kedvünk szerint.

A 14.1. táblázatban felsorolunk néhány gyakoribb, fontos megszakítást a teljesség igénye nélkül.

A következőkben néhány példán keresztül bemutatjuk a fontosabb szolgáltatások használatát.

Szöveget sokféleképpen lehet kiírni a képernyőre. Most az egyik DOS funkciót használjuk majd erre: A DOS-t az int 21h –val tudjuk segítségül hívni. A 4Ch sorszámú funkciót már eddig is használtuk a programból való kilépésre. 09h dollárjellel lezárt stringet ír ki a megfelelő karakterpozícióba. Billentyű kezelését az int 16h-n keresztül tudjuk elérni, a szolgáltatás számát itt is AH-ba kell rakni.:

```
MODEL SMALL
.STACK
ADAT SEGMENT
Kerdes DB "Tetszik az Assembly? $"
Igen DB "Igen.",0Dh,0Ah,'$'
Nem DB "Nem.",0Dh,0Ah,'$'
```

```
ADAT ENDS
KOD SEGMENT
ASSUME CS:KOD,DS:ADAT
@Start:
MOV AX,ADAT
MOV DS,AX
MOV AH,09h
LEA DX,[Kerdes]
INT 21h
@Vege:
MOV AH,09h
INT 21h
MOV AX,4C00h
INT 21h
KOD ENDS
END @Start
```

Egy assembly programban az INT 25h megszakítás segítségével lehet egy lemezről egy blokkot beolvasni. Ez nem karakteresen történik, hanem blokkokban, ehhez csak a forrás és cél címet, valamint a beolvasandó blokkok számát kell megadni. Egy floppy esetében egy blokk mérete 512 byte, így ennyi helyet kell lefoglalni hozzá az adatszegmensben.

```
.MODEL SMALL
Space EQU " "
.STACK
.DATA?
  Block DB 512 DUP(?)
.CODE
main proc
  MOV AX, Dgroup          ;DS beállítása
  MOV DS,AX
  LEA BX, BLOCK           ;DS:BX memóriacímre tölti a blokkot
  MOV AL,0                ;Meghajtó száma
  MOV CX,1                ;Beolvasott blokkok száma
  MOV DX,0                ;Lemezolvasás kezdőblokkja
  INT 25h                 ;Beolvasás
  POPF                    ;A veremben tárolt jelzőbitek törlése
  XOR DX,DX               ;Kiírandó adatok kezdőcíme DS:DX
  CALL write_block
  MOV AH,4Ch              ;Kilépés
main endp
```

Assembly és a pascal nyelv kapcsolata:

Pascalban használhatóak assembly kódok, közvetlenül a pascal kódba ágyazva. Az assembly kódot ASM jelöléssel kell nyitni, és END-el zárni. Assembly programmodulok futtatásakor nem kell verembe menteni az AX,BX,CX és DX regisztereket, de az SI, DI, BP, SP, CS, DS és SS-t igen.

Alapműveletek

```
Function Osszeg(A,B: Byte):Byte; Assembler;
ASM
  MOV AL,A
  ADD AL,B
END;
```

Hasonló módon oldható meg a kivonás (SUB), a szorzás (MUL, word típusú adatokkal) és az előjeles szorzás (IMUL). A függvény típusától függően kell bizonyos regiszterbe tölteni a visszatérési értékeket (eredményt):

- Byte, shortint (1 byte-os) értékek esetén az AL-be
- Word, integer (2 byte-os) értékek esetén AX-be
- Longint, pointer (4 byte-os) értékek esetén DX:AX regiszter párba
- Real (6 byte-os) értékek esetén DX:BX:AX regiszterekbe

Input-output műveletek

Karakter beolvasása és kiírása:

```
Procedure Kiir(C: Char); Assembler;
ASM
  MOV DL,C
  MOV AH,2                    ;Képernyőkiírás funkciókódja
  INT 21h
END;

Function Beolvas: Char; Assembler;
ASM
  MOV AH,1                    ;Beolvasás kódja
  INT 21h
END;
```

b. Számrendszerek közötti konverzió:

A konverzió átalakítást jelent, ami esetünkben annyit tesz, hogy közreadunk egy olyan módszert, amely segítségével egyetlen lépésben megoldható bármilyen szám felírása az ismert számrendszerekben, mert valójában minden ábrázolt szám ugyanabból a bitkombinációból áll. A számítógép nyolcas számrendszer esetében triádokat (három bitből álló csoport), míg a tizenhatos számrendszerben tetrádokat (négy bitből álló csoport) képez (a képzés minden esetben jobbról-balra halad).

A Konverzió Használata

1. lépés: mint említettük, egyetlen lépésben szeretnénk megoldani az átalakítást. Ehhez ismételten egy táblázatot használunk, de ennek már nem kettő, hanem több sora lesz. Ezekbe a sorokba fognak kerülni az egyes számrendszerek hatványai, illetve a maradék-képzéskor keletkezett értékek.
2. lépés: mielőtt a már elcsépelt 2001-es számot ismét elővennénk, tekintsük át ismét azon hatványokat melyeket korábban használtunk:

- Kettes (bináris) számrendszer: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512.
- Nyolcas (oktális) számrendszer: 1, 8, 64, 512.
- Tizenhatos (hexadecimális): 1, 16, 256.

3. lépés: észrevehető, hogy bizonyos hatványelemek fedik egymást. Ebből adódik, hogy a számrendszerek lefelé kompatibilisek, azaz a tizenhatos számrendszerből levezethető a nyolcas és a kettes, még a nyolcasból csak a kettes.
4. lépés: az áttekinthetőség kedvéért egy végső ellenőrzéssel nyugtázzuk munkánkat.

Példa A Konverzióra

1024	512	256	128	64	32	16	8	4	2	1	*
1	1	1	1	1	0	1	0	0	0	1	**
X	3	**X**	**X**	7	**X**	**X**	2	**X**	**X**	1	***
X	**X**	7	**X**	**X**	**X**	D	**X**	**X**	**X**	1	****

*: a legfelső sorba minden esetben a kettes számrendszer hatványai kerülnek, természetesen ügyelve a túlcsordulásra, azaz arra, hogy a hatványok ne lépjék túl az ábrázolandó szám értékét.
**: a kettes számrendszerbeli szám.
***: a nyolcas számrendszerbeli szám (arra a helyiértékre, amelyet az oktális számrendszer nem használ, megvastagított „**X**”-ek kerülnek).
****: a tizenhatos számrendszerbeli szám (arra a helyiértékre, amelyet a hexadecimális számrendszer nem használ, megvastagított „**X**”-ek kerülnek).

A Példa Ellenőrzése
$2001_{10} = 11111010001_2 = 3721_8 = 7D1_{16}$.

Egy Más Megközelítésű Ellenőrzés

Korábban már szóltunk a triádokról és a tetrádokról. Most ezeket alapul véve végezzük az ellenőrzést.
Tehát $2001_{10} = 11111010001_2$. Lássuk a lépéseket:

1. lépés: képezzünk bithármasokat (triádokat) jobbról-balra haladva, és mindegyik bithármas értékét számítsuk át tízes alapúra (az 1. 3. 1-es fejezetben tárgyaltuk a kettes és a tízes számrendszer közötti átváltást). Ha a képzés végén kiderül, hogy elfogytak a számok, nullákkal pótoljuk a hiányt (az esetleges pótlást esetünkben vastagítva fogjuk jelölni). Ezek után látható, hogy az alábbi bithármasok születtek, amelyeket azonnal váltsunk is át tízes alapúra. Íme az eredmény:
   - $001_2 = 1_{10}$
   - $010_2 = 2_{10}$
   - $111_2 = 7_{10}$
   - $\mathbf{0}11_2 = 3_{10}$

   Ha balról-jobbra (esetünkben alulról-felfelé) olvassuk a számokat, megkapjuk, hogy: $11111010001_2 = 3721_8$
2. lépés: most képezzünk bitnégyeseket (mivel a kettes számrendszerbeli szám pontosan 11 elemből áll, így ismételten szükséges egy nullával pótolnunk). Íme az eredmény:
   - $0001_2 = 1_{10}$
   - $1101_2 = 13_{10} = D_{16}$
   - $\mathbf{0}111_2 = 7_{10}$

Ha az előbbi módon összeolvassuk a számokat, az eredmény ismét pontos lesz: $11111010001_2 = 7D1_{16}$

c. Alulról-felfelé elemzés:

Alulról felfelé elemzéseknél a terminális szimbólumokból kiindulva haladnak a kezdőszimbólum felé. Ezek az elemzések egy vermet használnak, és a vizsgált módszerek léptetés-redukálás típusúak, azaz az elemző mindig egy redukciót próbál végezni, ha ez nem sikerül, akkor az input szimbólumsorozat következő elemét lépteti. Ezek az elemzések a legjobboldalibb levezetés „inverzét" adják. Az elemzés alapproblémája a mondatformák nyelének a meghatározása.
Szintaxisfa felépítésénél az elemzendő szimbólumsorozatból indulunk ki, megkeressük a mondatforma nyelét, amit helyettesítünk a hozzá tartozó nemterminális szimbólummal. A cél, hogy elérjük a grammatika kezdőszimbólumát, ami a szintaxisfa gyökérszimbóluma. A fa levelei pedig az elemzendő program terminális szimbólumai kell, hogy legyenek. Egy mondatforma legbaloldalibb egyszerű részmondatát a mondatforma nyelének nevezzük (egy mondatforma szintaxisfájában a legbaloldalibb, csak gyökérelemeket és leveleket tartalmazó részfa a mondatforma nyele).
Ha $A \rightarrow \alpha \in P$, akkor az $x A \beta$ mondatforma legbaloldalibb helyettesítése $x\alpha\beta$, azaz $xA\beta \rightarrow_{legbal} x\alpha\beta$. Ha $A \rightarrow \alpha$, akkor a βAx mondatforma legjobboldalibb helyettesítése $\beta\alpha x$, azaz $\beta Ax \rightarrow legjobb\ \beta\alpha x$. Példa: az ababb mondat elemzése a következő grammatika segítségével: G=({A,B,S},{a,b},P,S), S→AB, A→ab, B→aba.

Lépés	**Verem állapota**	**Állapot**	**Megjegyzés**
1.	#	.ababb	léptetés
2.	#a	a.babb	léptetés
3.	#ab	ab.abb	redukció (A→ab)
4.	#A	ab.abb	léptetés
5.	#Aa	aba.bb	léptetés
6.	#Aab	abab.b	redukció (A→ab)
7.	#AA	abab.b	léptetés
8.	#AAb	ababb.	nem lehet redukálni, visszalépés a 6.-ra
9.	#Aab	abab.b	nincs másik szabály, léptetés
10.	#Aabb	ababb.	nem lehet redukálni, visszalépés 3.-ra
11.	#ab	ab.abb	nincs másik szabály, léptetés
12.	#aba	aba.bb	redukció (B→aba)
13.	#B	aba.bb	léptetés
14.	#Bb	abab.b	léptetés
15.	#Bbb	ababb.	nem lehet redukálni, visszalépés 12.-re
16.	#aba	aba.bb	nincs másik szabály, léptetés
17.	#abab	abab.b	redukció (A→ab)
18.	#abA	abab.b	léptetés
19.	#abAb	ababb.	nem lehet redukálni, vissza a 17.-re
20.	#abab	abab.b	nincs másik szabály, léptetés
21.	#ababb	ababb.	nem lehet redukálni, visszalépés, de nincs hová