Лекция 9

Език Асемблер. Прекъсвания в компютърните системи

Вътрешни хардуерни прекъсвания

Вътрешните прекъсвания се генерират от събития, които се предизвикват при изпълнение на програмите. Този тип прекъсвания се управляват от хардуера на компютърната система и не е възможно те да бъдат променяни.
Пример за такова прекъсване е актуализацията на брояча на вътрешния часовник на компютъра. Хардуерът активира това прекъсване няколко пъти в една секунда за да поддържа системното време и дата в компютъра
Това прекъсване не може да се управлява чрез софтуера, защото реалното време се поддържа от електронни елементи в компютърната система.

Външни хардуерни прекъсвания

Външните прекъсвания се генерират от периферните устройства (клавиатура, мишка, принтер, разширителни карти и други). Не е възможно да се деактивират външни прекъсвания..
Тези прекъсвания не се изпращат директно към CPU, а към специално устройство, чиято задача е да управлява тези прекъсвания. Контролерът PIC8259A е създаден специално за тази цел, и той се управлява от CPU, използвайки няколко комуникационни линии, наричани IRQ линии.

Софтуерни прекъсвания

Софтуерните прекъсвания могат директно да се активират от програмата, като използват определен номер на прекъсването с помощта на инструкцията INT. Използването на такива прекъсвания позволява създаването на по-къси, ефективни и разбираеми програми.
Софтуерните прекъсвания могат да се разделят на две групи: DOS прекъсвания и BIOS прекъсвания.
Разликата между тези две групи е, че DOS прекъсванията са по-лесни за използване, но те са по бавни, защото трябва да реализират обръщения към BIOS функции. BIOS прекъсванията са по-бързи, но те имат недостатъка, че като част от хардуера (те се реализират на хардуерно ниво), имат много специфични характеристики и могат да зависят от конкретните хардуерни елементи (интегрални схеми).

1. Най-често използваните прекъсвания

1.1. 21H Interruption

Това прекъсване се използва за извикване на различни DOS функции.

Синтаксис: ***Int 21H***

Когато се работи с програмна среда TASM е необходимо да се укаже, че стойностите които се използват са в шестнадесетичен вид.
Това прекъсване има няколко функции, достъпът до които се осъществява като номера на функцията се задава в регистър AH.

- Функции за изобразяване на информация:

02H - изход към дисплвй; ***09H*** - изобразяване на последователност от данни (видео);
40H - писане в устройстъво / файл

- Функции за четене на информация от клавиатурата:

01H - въвеждане от клавиатура; ***0AH*** - въвеждане от клавиатура с използване на буфер; ***3FH*** - четене от устройство /файл

- Функции за работа с файлове:

Съществуват два метода за работа с файлове: FCB метод и метод с файлови манипулатори.

FCB Method

0FH – отваряне на файл (Open file); ***14H*** – четене с последователен достъп (Sequential reading) ; ***15H*** – писане с последователен достъп (Sequential writing); ***16H*** – създаване на файл (Create file); ***21H -*** четене с произволен достъп (Random reading); ***22H*** – писане с произволен достъп (Random writing).

Файлови манипулатори (Handles)

3CH – създаване на файл (Create file); ***3DH*** – отваряне на файл (Open file); ***3EH*** – затваряне на файл (Close file driver); ***3FH*** – четене от файл/устройство (Reading from file/device); ***40H*** – писане във файл/ устройство (Writing in file/device); ***42H –*** преместване на указателя при четене или писане във файл (Move pointer of reading/writing in file).

Синтаксис на някои функции на прекъсване 21H:

02H FUNCTION

Изобразява един символ върху дисплея. Номерът на функцията се задава в регистър AH, а символът, който трябва да се изобрази се въвежда в регистър DL:

AH = 02H

DL = Value of the character to display.

Тази функция изобразява символ, чийто шестнадесетичен код съответства на стойността зададена DL регистъра.

09H FUNCTION

Изобразява последователност от символи върху екрана:

AH = 09H

DS:DX = Address of the beginning of a chain of characters.

Функцията изобразява последователност от символи от зададения адрес, зададен в регистър DS:DX до поява на символ $, който се интерпретира като край на последователността.

40H FUNCTION

Писане в устройстъво или файл

AH = 40H

BX = Адрес на устройство, където се намира информацията

CX = Брой байтове, които да се изобразят

DS:DX = Адрес на началото на данните

Връщане на стойности:

CF = 0 ако няма грешка

AX = Брой изобразени байтове

CF = 1 ако има грешка

AX = Код на грешката

За изобразяване на информацията върху екрана в регистър BX трябва да се задае стойност 1, която се отнася за операционна система MS-DOS.

01H FUNCTION

Въвеждане от клавиатурата на 1 символ и изобразяването му върху екрана:

AH = 01H

Връщана стойност:

AL = Прочетената стойност

Прочетената стойност се съхранява в AL регистър.

0AH FUNCTION
Въвеждане от клавиатурата на символ и съхраняването им в буфер:
AH = 0AH
DS:DX = Адрес в паметта
BYTE 0 = Брой байтове в паметта
BYTE 1 = Брой прочетени байтове
от BYTE 2 до BYTE 0 + 2 = прочетени символи
Символите се четат и записват в предварително дефинирана област от паметта. Структурата на тази област предвижда в първия байт да се зададе броя на байтове, които трябва да се прочетат. Във втория байт се задава броя на байтове прочетени до момента и от третия байт се записват прочетените байтове.
Когато всички предвидени байтове са записани в паметта се подава звуков сигнал и другите символи се игнорират. За да завърши въвеждането на символи трябва да се натисне [ENTER].

3FH FUNCTION
Четене на информация от устройство или файл:
AH = 3FH
BX = Адрес на устройство, където се намира информацията
CX = Брой байтове, които да се изобразят
DS:DX = Адрес на началото на данните
Връщани стойности:
CF = 0 ако няма грешка AX = брой прочетени байтове.
CF = 1 ако има грешка AX ще съдържа кода на грешката.

1.2. 10h Interruption
Това прекъсване се използва за извикване на различни BIOS video функции
Синтаксис: ***Int 10H***
Това прекъсване има няколко функции, всички за контролиране на video изхода. За да се използва определена функция е необходимо номерът на функцията да бъде записан в ***Ah*** регистъра.
Някои от функциите на 10h прекъсването
02H Function, избира позицията на показалеца.
09H Function, записва атрибут и символ на мястото, където е позициониран показалеца.
0AH Function, записва симвил на мястото, където е позициониран показалеца.

02h Function
Премества показалеца върху екрана на компютъра:
AH = 02H
BH = Video page, където е позициониран показалеца.
DH = row (ред)
DL = Column (колона)
Местоположението на показалеца се определя от координати, започвайки от позиция 0,0 до позиция 79,24. Това означава, че левия горен ъгъл на екрана се задава с координати 0,0, а десния долен ъгъл с координати 79,24. Следователно, числените стойности зададени в регистри DH и DL представляват номер на ред и колона върху решетката на екрана.

09h Function
Тази функция показва символ няколко пъти с дефиниран атрибут:
AH = 09H
AL = символ, който трябва да се покаже
BH = Video page, където е позициониран показалеца.
BL = използван атрибут
CX = брой на повторенията на изобразяване на символа
Тази функция показва символ няколко пъти, използвайки стойност (брой) зададен в регистър ***CX***, но без промяна на положението на показалеца върху екрана

0Ah Function
Показва символ върху екрана в действителната позиция на показалеца:
AH = 0AH
AL = символ за изобразяване на екрана
BH = Video page, където е позициониран показалеца.
BL = използван цвят (в графичен режим).
CX = брой на повторенията
Основната разлика от предишната функция е, че не се извършва модификация на атрибутите и не се променя позицията на показалеца.

0EH Function
Изобразява се символ върху екрана на компютъра в съответствие с позицията на показалеца.
AH = 0EH
AL = символ за изобразяване на екрана
BH = Video page, където е позициониран показалеца.
BL = използван цвят (в графичен режим).

1.3. 16H interruption
Ще бъдат представени две функции за това прекъсване. Това са функции за четене на информация от клавиатурата.
00H Function
Използва се за прочитане на символ от клавиатурата. В регистър ***AH*** се задава номера на функцията:
AH = 00H
Връщани стойности:
AH = Scan code от клавиатурата
AL = ASCII код на символа, който се прочита от клавиатурата
Когато се използва това прекъсване изпълнението на програмата се прекъсва до натискане на съответен бутон от клавиатурата. В регистър AH се съхранява *scan* кода, а ако това е ASCII код той се съхранява в регистър AL. Scan кодът се използва за въвеждане на символи без ASCII код (контролни символи).

01h function
Тази функция прочита състоянието на клавиатурата:
AH = 01H
Връщани стойности:
Ако флаг регистъра е нула, означава че има информация в буфера, а в противен случай – няма информация в паметта на буфера.

1.4. 17H Interruption

Това прекъсване се използва за управление на принтер.
Синтаксис: ***Int 17H***
Използва се за отпечатване на символ върху принтера, задава състояние на принтера и чете състоянието на принтера.
Основни функции на прекъсване ***16h:***
00H Function, отпечатва символ по неговия ASCII код
01H Function, задава състояние на принтера
02H Function, показва състоянието на принтера

00H Function
Отпечатва символ върху принтера.
Състояние на регистрите:
AH = 00H
AL = символ за отпечатване
DX = използван порт.
Връщани стойности:
AH = състояние на принтера.
Използваният порт се задава в регистър DX, като стойностите са: ***LPT1 = 0, LPT2 = 1, LPT3 = 2 ...***
Състоянието на принтера се кодира бит по бит както следва:

Бит	1/0	Значение
0	1	Изтекло време за изчакване
1	-	
2	-	
3	1	входно/изходна грешка
4	1	Избран принтер
5	1	липсва хартия
6	1	Връзка установена
7	1	Принтер готов за работа

Битове 1 и 2 не се използват

01h Function
Задава състояние на принтера
Състояние на регистрите:
AH = 01H
DX = използван порт.
Връщани стойности:
AH = състояние на принтера.
Състоянието на принтера се кодира бит по бит както и е както горната функция.

02h Function
Прочита състояние на принтера.
Състояние на регистрите:
AH = 02H
DX = използван порт.
Връщани стойности:
AH = състояние на принтера.
Състоянието на принтера се кодира бит по бит както и е както горната функция.

2. **Начини за работа с файлове**

Съществуват два начина за работа с файлове. Първият начин е чрез използване на контролни блокове или "FCB" метод и вторият е с използване на комуникационни канали, известни още като файлови ‘манипулатори‘.
Първият начин е бил използван още с разработване на CPM операционна система, предшественик на DOS, като по този начин се осигурява съвместимост с много стари файлови системи. Освен това, този начин позволава в определен момент да има неограничен брой отворени файлове.
Въпреки предимствата на FCB метода, използването на методът на комуникационните канали е по-просто и позволява по-лесно обработване на грешките.

Метод **FCB**
Съществуват два типа FCB, нормален, чиято дължина е 37 байта и разширен тип с дължина 44 байта. Тук ще бъде разгледан нормален тип FCB файлове.
FCB файловете се композират от програмиста с използване на информация от операционната система. Когато се използват този тип файлове е възможно да се работи само в текущата директория (папка).
FCB се формират от следните полета:

Позиция	**Дължина**	**Значение**
00H	1	Индекс на диска
01H	8	Наименование на файла
09H	3	Разширение на файла
0CH	2	Номер на блока
0EH	2	Размер на регистъра
10H	4	Размер на файла
14H	2	Дата на създаване на файла
16H	2	Час на създаване на файла
18H	8	Резервни байтове
20H	1	Текущ регистър
21H	4	Произволен регистър

За избора на диск се използва следният формат: диск A = 1; диск B = 2; и т.н. Ако е зададена стойност 0 се използва текущият диск.
Името на файла трябва да бъде подравнен вляво и ако е необходимо се добавят пробели до края на името (8 байта). Същото се отнася и за разширението на файла.
Текущ блок и текущ регистър указват кой регистър ще бъде достъпен за операции за четене или записване на данни във файла. Блокът е група от 128 регистъра. Първият блок се номерира с 0. Първият регистър е регистър 0, следователно последният регистър е с номер 127.
Работата с файловете се осъществява с помощта на файловите функции на прекъсване ***21H,*** които бяха изброени по-горе.

Отваряне на файл
За отваряне на FCB файл се използва ***0FH*** функция на ***21H*** прекъсване.
Състояние на регистрите:
AH = 0FH
DS:DX = Указател за FCB
Връщани стойности:
AL = 00H ако няма проблеми или ***0FFH*** – ако е възникнал проблем.
Името и разширението на файла трябва да бъдат инициализирани преди отварянето на файла. Регистърът DX трябва да съдържа адреса на блока за управление на файла

(FCB). Ако връщаната стойност е ***FFH*** в регистър ***AH*** означава, че файлът не е намеран.
Ако файла се отвори нормално DOS инициализира текущия блок като 0, размера на регистъра 128 байта и размера и датата на създаване ха файла се попълват с информация предоставена от операционната система.

Създаване на нов файл
За създаване на FCB файл се използва ***16H*** функция на ***21H*** прекъсване.
Състояние на регистрите:
AH = 1FH
DS:DX = Указател за създадена FCB
Връщани стойности:
AL = 00H ако няма проблеми или ***0FFH*** – ако е възникнал проблем.
Регистър ***DX*** трябва да задава адрес на контролна структура, която да дефинира логическа единица, име и разширение на файл.

Достъп за последователен запис
Трябва да се дефинира областта за трансфер на данните и да се използва ***1AH*** функция на прекъсване ***21H***. Функцията ***1AH*** не връща състояние на диска или операцията, а активира функция ***15H***, която се използва за запис върху диск.

Достъп за последователно четене
Трябва да се дефинира областта за трансфер на данни. За да се извърши достъп за последователно четене се използва функция 14H на прекъсване 21H. ***AL*** регистър връща състоянието на операцията. Ако ***AL*** съдържа стойност от 1 или 3 означава, че е достигнат края на файла. Стойност 2 означава, че FCB е грешно структуриран. В случай че няма грешка стойността на AL регистъра е 0.

Произволен достъп за четене и писане.
Функции ***21H*** и ***22H*** на прекъсване ***21H*** се използват за реализиране на произволен достъп за четене и запис. Регистър за произволен достъп и текущия блок се използват за изчисляване на относителната позиция на регистъра за четене и запис.
Регистър ***AL*** връща същата информация както при последователното четене или запис.

Затваряне на файл
За затваряне на файл се използва функция 10H на прекъсване 21H. Ако след изпълнение на тази функция регистър AL съдържа FFH стойност означава, че на файла е сменена позицията, дискът е сменен или има грешка при достъпа до файла.

Комуникационни канали
Използването на манипулатори за управление на файловете осигурява възможност за програмиране, без да се държи сметка на детайлите по организация на файловете. Това се прави от операционната система.
Лесното използване на манипулаторите за организация на файловете се изразява в това, че програмиста трябва да зададе само името на файла и номера на манипулатора, а цялата друга информация се доставя от DOS.
Когато се използва този метод не е необходимо да се организира последователен или произволен достъп, файловете просто се разглеждат като последователност (верига) от байтове.

Функциите за използване на файлови манипулатори са подобни на функциите за използване на FCB метода.

3. Макроси и процедури

- **Процедури**

Дефиниция на процедура

Процедурата е съвкупност от инструкции, към които може да се насочва потока на изпълнение на инструкциите от програмата и след като се изпълнят инструкциите от процедурата управлението се предава обратно на следващата инструкция от потока на програмата.

Процедурите позволяват съставянето на логични и лено модифицируеми програми.

Когато се извика процедура, адресът на следващата команда от потока на програмата се записва във стека и след изпълнението на процедурата се прочита този адрес от стека и изпълнението на програмата продължава с тази инструкция.

Синтаксис на процедурите

Съществуват два типа процедури, вътрешносегментни, които се намират в текущия сегмент на инструкциите и междусегментни, които могат да бъдат записани в други сегменти от паметта.

Когато се използват вътрешносегментни процедури стойността на ***IP*** регистъра (указател за инструкциите) се съхранява в стека, а когато се използва междусегментна процедура се съхранява стойността на ***CS:IP***

За извикване на процедура се използва следната директива (команда):

CALL NameOfTheProcedure

Основни части на процедурата са:

- Декларация на процедурата
- Код на процедурата
- Return директива
- Завършване на процедурата

Например една прцедура, която сумира два байта съхранявани в регистри ***AH*** и ***AL*** и съхраняване на сумата в регистър ***BX*** би изглеждала по следния начин:

Adding Proc Near ; *Декларация на процедурата*
Mov Bx, 0 ; *Код на процедурата*
Mov B1, Ah
Mov Ah, 00
Add Bx, Ax
Ret ; *Return директива*
Add Endp *; Край на процедурата*

Декларацията задава името на процедурата '***Adding***', като се използва служебната дума '***Proc***'. Думата '***Near***' индикира, че процедурата е вътрешносегментна. Директивата ‚***Ret***' зарежда ***IP*** адресът съхранен в стека за да може изпълнението на програмата да продължи от командата преди която е извикана процедурата. Накрая ‚***Add Endp***' директивата указва за края на процедурата.

За да се създаде междусегментна процедура трябва да се използва вместо думата ‚***Near***' служебната дума ‚***FAR***'.

Извикването на процедурата се извършва по следния начин:

Call Adding

- **Макроси**

Макросите осигуряват по-голяма гъвкавост при разработване програми в сравнение с процедурите.

Дефиниция на макроси
Макросите са група повтарящи се инструкции в програмата, които са кодирани само веднъж и могат да се използват многократно. Основната разлика от процедурите е, че в макросите могат да се използват параметри (за процедурите това е възможно само за TASM – там се използват други програмни езици, които позволяват това).
Когато се изпълнява даден макрос, параметрите се заместват с имена на действителни променливи или стойности в момента на извикването им. Може да се каже, че процедурите са едно разширение на програмите, докато макросите са модули със специфична функционалност и могат да се използват в различни програми.
Друга разлика между процедури и макроси е начинът на извикване. Извикването на процедурата изисква специална директива (Call), докато макросът се използва като инстукция на Асемблер.

TOP

Синтаксис на Макрос
Основните части на един макрос са:
Декларация на макроса
Код на макроса
Директива за край на макроса
Декларацията на макроса се задава по следния начин:
NameMacro MACRO [parameter1, parameter2...]
NameMacro е името на макроса, ***MACRO*** е служебна дума за обозначаване на декларация на макрос и в квадратни скоби е зададен списък с параметри – той може да има различен брой параметри. Макросът може да бъде и без параметри.
ENDM е директива за край на макрос.
Като пример е използван макрос за разполагане на показалеца в определена позиция на екрана:

```
Position MACRO Row, Column
PUSH AX
PUSH BX
PUSH DX
MOV AH, 02H
MOV DH, Row
MOV DL, Column
MOV BH, 0
INT 10H
POP DX
POP BX
POP AX
ENDM
```

За използването на макроса е достатъчно на се запише името на макроса като обикновена инструкция на Асемблер:

```
Position 8, 6
```

Библиотеки с макроси
Една от добрите страни на използване на макроси е създаването на библиотеки, като групи от макроси, които могат да се включват в различни асемблерски програми. Създаването на тези библиотеки е много просто. Трябва просто да се създаде текстов файл със създадените макроси. За извикване на определен макрос е необходимо да се използва инструкцията ***Include NameOfTheFile***, като част от разработваната програма в началото преди декларацията за модела на паметта.
Ако е създаден файл с макроси, например MACROS.TXT, необходимите инструкции за използване на макросите са:

```
;Beginning of the program
```

```
Include MACROS.TXT
.MODEL SMALL
.DATA
;The data goes here
.CODE
Beginning:
;The code of the program is inserted here
.STACK
;The stack is defined
End beginning
;Our program ends
```