Министерство образования Республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра электронно-вычислительных машин

Лабораторная работа №6

«Функции BIOS»

Выполнил:

студент группы 724402

Боешко Д.Ю.

Проверил:

к.т.н., доцент Селезнёв И.Л.

Минск  
 2019

1. **Цель работы**

Изучить функции BIOS и организацию ввода-вывода через прерывания BIOS.

1. **Описание алгоритма**
   1. Вызвать прерывание INT 11h для получения конфигурации   
      компьютера.
   2. Вызвать прерывание INT 12h для получения объема основной оперативной памяти компьютера.
   3. Вызвать прерывание INT 16h для ввода текста.
   4. Вызвать прерывание INT 1Аh для считывания и установки даты и времени с часов реального времени.
2. **Теоретические сведения**

BIOS (Basic­Input/Output­System — базовая­ система ­ввода/вывода) — это программа, предназначенная для первоначального запуска компьютера, настройки оборудования и обеспечения функций ввода/вывода. BIOS записывается в микросхему flash-памяти, которая расположена на системной плате.

**Системный модуль ROM BIOS (System ROM BIOS)** обеспечиваетпрограммную поддержку стандартных устройств PC, конфигурирование аппаратных средств, их диагностику и вызов загрузчика операционной системы. Системный модуль ROM BIOS в значительной степени привязан к конкретной реализации системной платы, поскольку именно ему приходится программировать все микросхемы чипсета системной платы.

*Функции BIOS разделяются на следующие группы:*

1. Инициализация и начальное тестирование аппаратных средств – POST

**Самоконтроль при включении питания** (Power-On Self-Test - POST). POST представляет собой встроенную диагностическую программу, которая проверяет наличие и правильность функционирования аппаратных средств до производства BIOS фактической загрузки. В ходе загрузки производится и дополнительное тестирование, например, тест памяти, ход которого отображается на экране;

1. Настройка и конфигурирование аппаратных средств и системных ресурсов – CMOS Setup;
2. Автоматическое распределение системных ресурсов - PnP BIOS

BIOS является одним из трех компонентов системы, взаимодействие которых необходимо для реализации в РС средств Plug and Play (PnP). Здесь аппаратные средства РС, BIOS и операционная система, работая в тесном взаимодействии, автоматически идентифицируют и конфигурируют использование ресурсов аппаратными устройствами. Технология PnP сокращает число конфликтов ресурсов, устраняет использование перемычек и ручную настройку драйверов. BIOS играет при этом ключевую роль, так как

фактически именно он идентифицирует и конфигурирует карты расширения,

а также передает информацию о конфигурировании операционной системе;

1. Идентификация и конфигурирование устройств PCI - PCI BIOS

Обычно шина большей частью PCI настраивается автоматически, но параметры этой секции могут пригодится, когда необходимо реализовать конкретное поведение шины или карт расширения. Для настройки параметров данной секции требуются глубокие знания аппаратных средств компьютера, поэтому для большинства пользователей обычно подходят принимаемые по умолчанию значения параметров. Рекомендуется также иметь в компьютере средства автоматического конфигурирования;

1. Начальная загрузка (первый этап загрузки операционной системы) - Bootstrap Loader;
2. Обслуживание аппаратных прерываний от системных устройств (таймера, клавиатуры, дисков) –BIOS Hardware Interrupts;
3. Отработка базовых функций программных обращений (сервисов) к системным устройствам - ROM BIOS Services;
4. Поддержка управляемости конфигурированием - DMI BIOS;
5. Поддержка управления энергопотреблением и автоматического конфигурирования - АРМ и ACPI BIOS.

Все эти функции (или их часть) исполняет системный модуль BIOS, хранящийся в микросхеме ПЗУ или флэш-памяти на системной плате. Большинство сервисных функций выполняется в 16-битном режиме, хотя некоторые новые функции могут иметь и альтернативные вызовы для 32-битного исполнения.

На экране загрузки BIOS можно найти следующую информацию (Рисунок 3.1).

1. **Производитель BIOS и номер версии.**

**Дата BIOS:** ДатаBIOSпомогает определить его возможности,так какдаты введения некоторых возможностей хорошо известны.

**Серийный номер BIOS:** Обычно отображается внизу экрана.Так какBIOS адаптированы под конкретную материнскую плату, серийный номер можно использовать для определения конкретной материнской платы и версии BIOS.

1. **Тип процессора (CPU):** Обычно это общее семейство процессоров,например, "Pentium" или "Pentium Pro" и т.д. Новые BIOS прямо распознают Intel-совместимые процессоры, а старые могут вывести "Pentium", даже когда процессор другой. Иногда BIOS отображает "Pentium-S", показывая, что процессор поддерживает расширенные средства управления мощностью (System Management Mode - SMM). Сейчас эти средства имеются почти во всех современных процессорах.
2. **Тип, ёмкость и конфигурация памяти:** в большинстве новых РСсообщается, сколько банков памяти обнаружено и какая технология памяти применяется. Например, можно увидеть на экране "EDO DRAM at Bank 1" или "FP: 0" (FPM DRAM) или что-то аналогичное.
3. **Накопители IDE/ATA:** Для каждого установленного накопителясообщается его емкость и используемые им режимы обращения. Некоторые BIOS сообщают также производителя накопителя, а большинство современных BIOS обнаруживает и показывает накопитель IDE (ATAPI) CD-ROM.
   * + - **5. Клавиша программы настройки:** Клавиша или клавиши,которыенеобходимо нажать для входа в программу настройки BIOS. Обычно это клавиша Del или F2.



Рисунок 3.1 – Сообщение процедуры POST

На экране загрузки BIOS можно найти следующую информацию:

**Сервисы и другие векторы прерываний BIOS**

При инициализации таблицы прерываний BIOS отвечает за корректное заполнение части векторов, имеющих отношение к аппаратным средствам компьютера и сервисам BIOS. На некоторые из них могут быть просто установлены заглушки: вектор ссылается на код обработчика, содержащего единственную инструкцию возврата из прерывания - I RET. BIOS инициализирует векторы прерываний различных назначений:

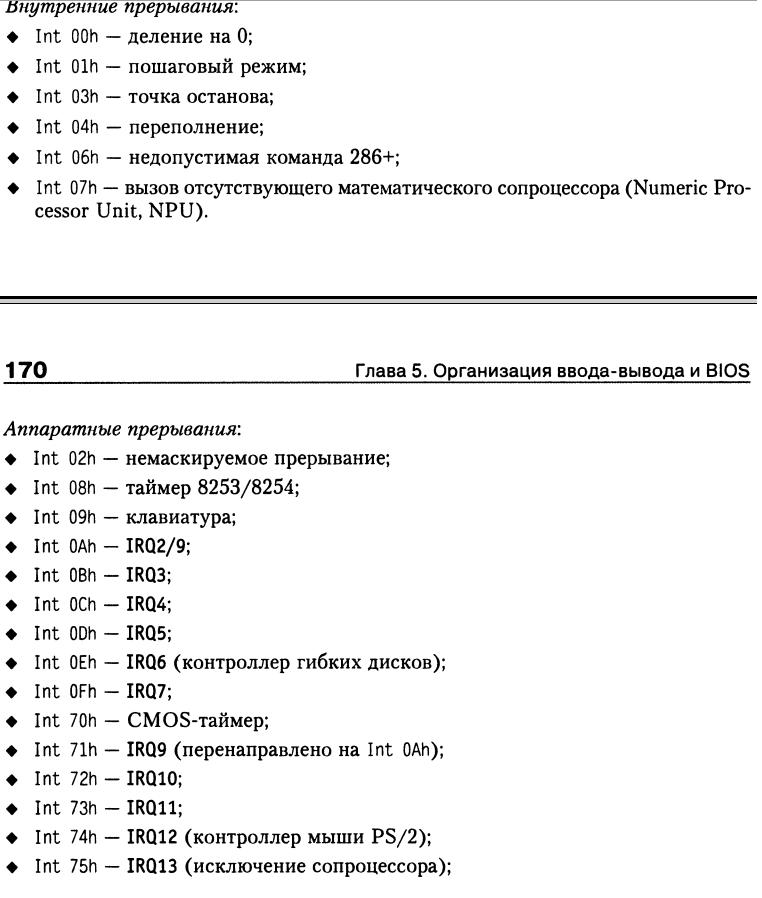
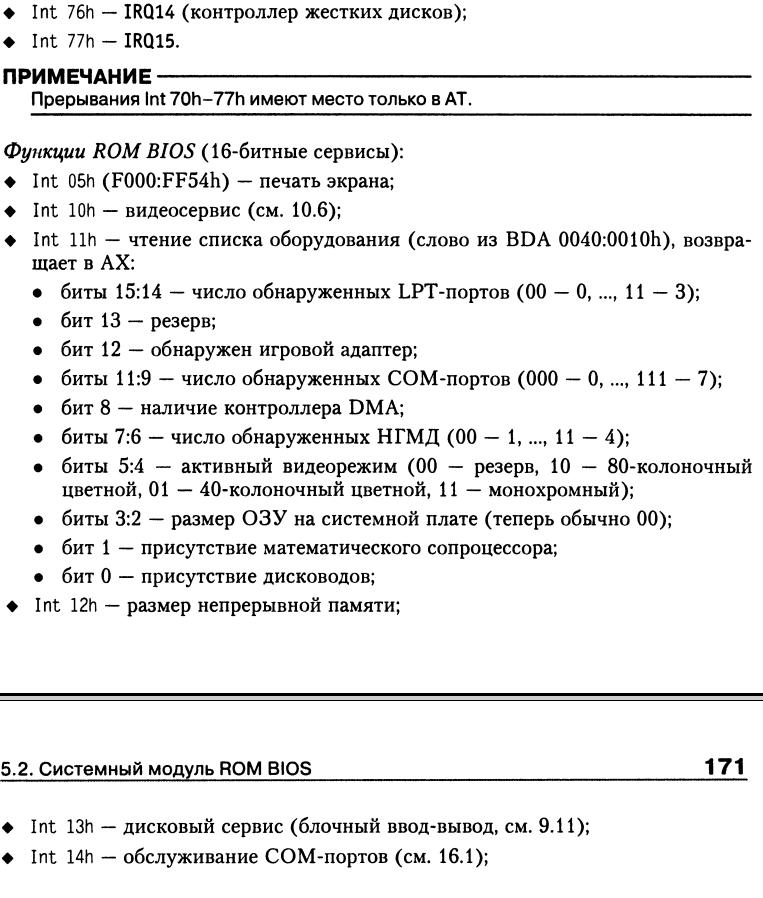


Рисунок 3.2 - Векторы прерываний BIOS



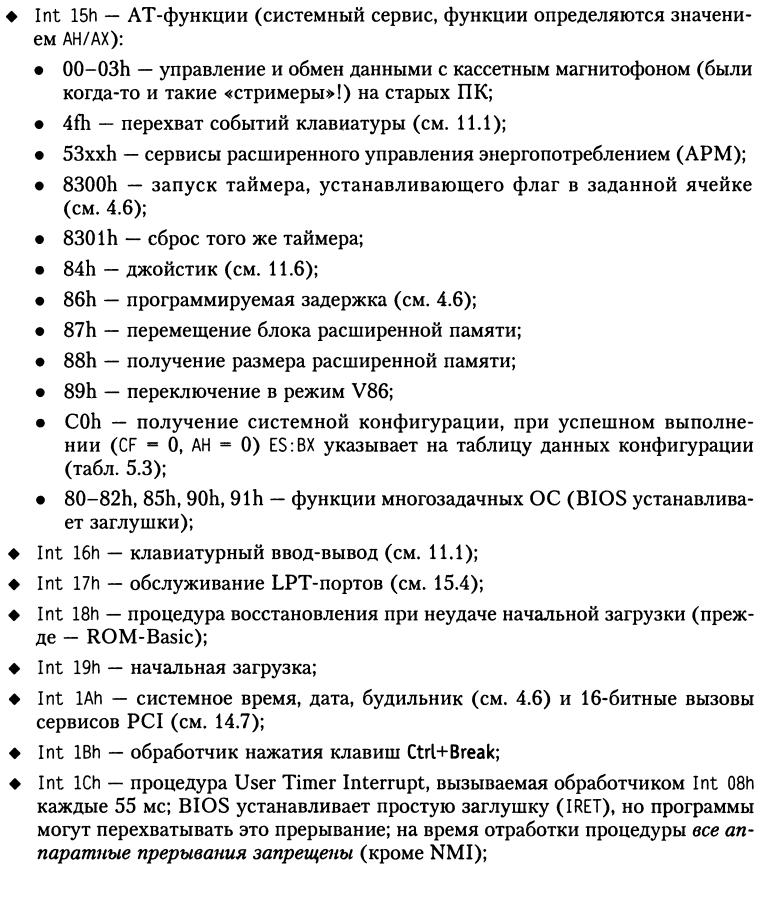


Рисунок 3.3 - Векторы прерываний BIOS

**INT 11h - Получить список оборудования.**

Прежде чем пытаться работать с каким-либо устройством ввода/вывода, следует убедиться в том, что оно есть в составе оборудования компьютера. В процессе инициализации тестовые модули, находящиеся в BIOS, динамически определяют состав аппаратного обеспечения машины и записывают конфигурацию системы в специально отведенную для этого ячейку памяти. Программа, вызывая прерывание INT 11h, получает в регистре **AX** содержимое этой ячейки. Каждый бит в слове конфигурации отвечает за соответствующее устройство.

Анализируя слово конфигурации, программа может узнать, входят ли в состав оборудования компьютера дискеты и если входят, то сколько

дисководов имеется в наличии, присутствует ли арифметический сопроцессор, какой начальный режим дисплейного адаптера используется, сколько в системе принтеров, адаптеров последовательного интерфейса RS232, подключен ли игровой адаптер (джойстик)?

Обычно прикладная программа не работает сама с аппаратурой, а пользуется услугами операционной системы. При обращении к стандартной аппаратуре через операционную систему программа пользователя получит признак ошибки, если запрашиваемое устройство отсутствует. Программы, составленные на языке программирования Си при использовании библиотеки эмуляции арифметического сопроцессора, сами определяют, имеется сопроцессор или нет, и не пытаются пользоваться отсутствующим устройством. Операции сопроцессора эмулируются центральным процессором, и программа просто работает медленнее.

Но если программа обращается непосредственно к портам ввода/вывода отсутствующего устройства, это может привести в лучшем случае к зависанию системы.

**INT 12h - Получить размер основной памяти.**

Сказанное выше справедливо и по отношению к оперативной памяти. Для работы некоторых программ требуется достаточное количество памяти. Прерывание INT 12h возвращает в регистре **AX** количество имеющихся блоков памяти размером в один килобайт. Анализируя эту величину, программы могут при нехватке памяти либо вывести на экран соответствующее сообщение и отказаться от работы, либо изменить алгоритмы работы, организовав, например, "виртуальную" память на диске или просто записывая в файл промежуточные результаты. Если Ваш компьютер оборудован расширенной памятью (адресное пространство этой памяти находится выше границы в 1 мегабайт), размер этой памяти в килобайтах можно узнать, вызвав прерывание INT 15h со значением регистра **AX**, равным 8800h.

**INT 16h ­­­­­­- Обслуживание клавиатуры.**

Обработчик прерывания INT 16h выполняет несколько функций, связанных с обслуживанием клавиатуры. Мы не будем сейчас перечислять эти функции, они будут подробно описаны в главе, посвященной клавиатуре. С помощью функций обслуживания клавиатуры можно выполнить ввод кода нажатой клавиши как с ожиданием нажатия, так и без ожидания. В последнем случае функция сразу после вызова возвращает код нажатой клавиши или признак того, что никакая клавиша не нажималась.

Заметим, что символы, введенные с клавиатуры, помещаются в специальный клавиатурный буфер. Функция ввода символа без ожидания нажатия на клавишу проверяет состояние буфера - есть в нем символы, или нет. Если в буфере есть символы, первый помещенный в буфер символ возвращается программе. Этот символ затем может быть считан функцией ввода с ожиданием нажатия - фактически ожидания при этом не будет.

Для очистки буфера клавиатуры также можно использовать пару описанных выше функций: сначала программа проверяет пуст ли буфер, и,

если он не пуст, считывает символ. Считанный символ никуда не помещается (теряется). После считывания символа программа опять проверяет содержимое буфера и так до тех пор, пока клавиатурный буфер не окажется пустым.

Для машин класса не ниже AT обработчик прерывания INT 16h выполняет и другие функции: установку задержки, запись символов в буфер клавиатуры, обслуживание расширенной клавиатуры.

**Обслуживание дисковой подсистемы.**

Прерывание INT 13h предназначено для обслуживания жестких и флоппи-дисков. Многочисленные функции прерывания INT 13h выполняют все операции по вводу/выводу на диски. Мы сделаем обзор только самых важных функций, остальные будут рассмотрены в книге 3, посвященной файловой системе.

00h - Сброс дисковой системы. Эта функция выполняет установку в исходное состояние всей дисковой системы или выбранного дискового устройства. Используется обычно перед началом работы с устройством.

01h - Получить состояние дисковой системы. Эта функция позволяет проверить результат выполнения предыдущей операции. Если операция завершилась аварийно, при помощи этой функции можно определить код ошибки.

02h/03h - Чтение/запись секторов. Выполняется чтение секторов в оперативную память компьютера или запись информации из памяти в сектора диска. Сектор задается для выбранных устройства, дорожки и головки. Программа должна также задать количество читаемых/записываемых секторов.

04h - Проверка секторов. Функция проверяет сектора на правильность циклической контрольной суммы, CRC (Cyclic Redundancy Check); записи содержимого секторов в память не происходит.

**Другие функции прерывания INT 13h.**

Среди других функций прерывания INT 13h - форматирование дорожки, позиционирование головки на заданную дорожку диска, тестирование и предварительная установка диска, запуск диагностики контроллера и многое другое. Описание этих функций мы отложим до глав, посвященных файловой системе.

**Вывод на принтер (параллельный порт).**

BIOS содержит простейшую поддержку принтера - три функции прерывания INT 17h. Это функция 01h - инициализация принтера, 02h - опрос состояния принтера и 00h - вывод символа на принтер. Поскольку к персональному компьютеру можно подключить несколько последовательных портов, при обращении к принтеру следует указывать номер порта.

**Обслуживание последовательного порта связи**

Функции прерывания INT 14h обслуживают порт последовательной передачи данных RS232. С помощью этих функций можно задавать формат и скорость передачи данных, определять состояние портов и, конечно, выполнять побайтную передачу данных.

**INT 1Ah ­­­­­­ – Работа с системными часами.**

Этот сервис предоставляет доступ к системным часам. PC BIOS работает со "счетчиком тиков" - числом 55-мс интервалов, прошедших с момента включения или сброса PC. AT BIOS предоставляет также доступ к значениям часов реального времени, которые постоянно обновляются независимо от работы процессора и хранятся в CMOS-памяти компьютера AT.

AH сервис

--- ----------------------------------------------------------------------------

00H читать часы (счетчик тиков)

выход: CX, DX = счетчик тиков с момента сброса. CX - старшая часть значения.

AL = 0, если таймер не переполнялся за 24 часа с момента сброса.

замечание: часы обновляются каждые 1193180/65536 (ў 18.2) тиков в секунду.

тиков в секунде 18

тиков в минуте 1092

тиков в часе 65543

тиков в сутках 1573040

--- ----------------------------------------------------------------------------

01H установить часы (счетчик тиков)

вход: CX, DX = счетчик тиков. CX - старшая часть значения.

--- ----------------------------------------------------------------------------

02H ¦AT¦ читать время из "постоянных" (CMOS) часов реального времени

выход: CH = часы в коде BCD (пример: CX = 1243H = 12:43)

CL = минуты в коде BCD

DH = секунды в коде BCD

выход: CF = 1, если часы не работают

--- ----------------------------------------------------------------------------

03H ¦AT¦ установить время на "постоянных" (CMOS) часах реального времени

вход: CH, CL = часы, минуты в коде BCD

DH = секунды в коде BCD

DL = 1 для опции "единиц светового дня"

--- ----------------------------------------------------------------------------

04H ¦AT¦ читать дату из "постоянных" (CMOS) часов реального времени

выход: CH = столетие в коде BCD (пример: CX = 1987H = 1987)

CL = год в коде BCD

DH = месяц в коде BCD (пример: DX = 0312H = 12-е марта)

DL = день в коде BCD

выход: CF = 1, если часы не работают

--- ----------------------------------------------------------------------------

05H ¦AT¦ установить дату на "постоянных" (CMOS) часах реального времени

вход: CH, CL = столетие, год в коде BCD

DH, DL = месяц, день в коде BCD

--- ----------------------------------------------------------------------------

06H ¦AT¦ установить сигнал часов реального времени. В указанное время вызывается

пользовательская программа по вектору прерывания INT 4aH. Лишь один сигнал

может быть активен в каждый момент времени.

вход: CH, CL = часы, минуты в коде BCD

DH = секунды в коде BCD

выход: CF = 1, если часы не работают или сигнал уже активен

--- ----------------------------------------------------------------------------

07H ¦AT¦ сбросить сигнал часов реального времени. это позволяет вам отменить

один сигнал перед установкой другого.

Функции прерывания INT 1Ah обслуживают часы, имеющиеся в каждом компьютере. С их помощью вы можете установить время и дату, опросить текущее состояние часов. Вы можете работать с часами реального времени, которые имеются на машинах класса не ниже AT.

Для AT можно установить на заданное время "будильник" - в нужный момент будет вызвано прерывание "будильника" с номером 4Ah. Обработчик прерывания INT 4Ah может подать звуковой сигнал или вывести на экран предупреждающее сообщение.

**Перезагрузка операционной системы**

Вызов прикладной программой прерывания INT 19h приведет к перезагрузке операционной системы.

**Системный сервис для машин класса AT.**

Прерывание INT 15h использовалось в компьютерах IBM PC и IBM PC Jr для управления кассетным накопителем на магнитной ленте (функции 0-3). Для машин класса AT и более высокого класса прерывание INT 15h имеет и другое назначение. С его помощью обслуживается расширенная клавиатура, выполняется программная задержка, задаваемая в микросекундах, обслуживается расширенная память. Кроме того, одна из функций прерывания INT 15h переводит процессор 80286 или 80386 в защищенный режим. Заметим, что вернуть процессор обратно в реальный режим можно только сигналом начального сброса. Это же относится и к арифметическому сопроцессору 80287.

Функция C0h прерывания INT 15h выдает дополнительные сведения о конфигурации аппаратных средств компьютера.

Системный модуль BIOS должен обслуживать по вышеуказанным функциям все компоненты, установленные на системной плате: процессор, контроллер памяти (ОЗУ и кэш), стандартные архитектурные компоненты (контроллеры прерываний и DMA, системный таймер, системный порт, CMOS RTC), контроллер клавиатуры, а также набор стандартных периферийных контроллеров и адаптеров, даже если они и не установлены на системной плате. В этот набор входят графические адаптеры CGA и MDA, порты СОМ и LPT, контроллер НГМД, диски ATA (теперь уже обязательно двух каналов). Если на системной плате установлены дополнительные компоненты, например, контроллер SCSI, графический адаптер SVGA, адаптер локальной сети, то их поддержка тоже должна быть в системном модуле BIOS.

1. **Листинг кода программы**

// 724402-1 Boeshko D.U. LR6 "Funktsii BIOS"

#include <bios.h>

#include <time.h>

#include <stdlib.h>

#include <stdio.h>

#include <memory.h>

#include <conio.h>

#include <dos.h>

int bcd1bin(char \*bcd) //BCD to BIN -> 1 byte

{

return (((\*bcd) & 0x0f) + 10 \* (((\*bcd) & 0xf0) >> 4));

}

int bcd2bin(char \*bcd) // BCD to BIN -> 2 byte

{

return (bcd1bin(bcd) + 100 \* bcd1bin(bcd + 1));

}

int bin1bcd(int bin, char \*bcd) // BIN to BCD -> 1 byte

{

int i;

i = bin / 10;

\*bcd = (i << 4) + (bin - (i \* 10));

return i;

}

char\* tobin(char\* str, unsigned char t)

{

int mask, i;

for (i = 0, mask = 0x80; mask != 0; mask >>= 1, i++)

{

str[i] = t & mask? '1' : '0';

}

str[8] = '\0';

return str;

}

char\* tohex(char\* str, unsigned char t)

{

char hex[16] = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};

str[2] = '\0';

str[1] = hex[t & 0xF];

str[0] = hex[t >> 4];

return str;

}

void showWord(unsigned int uword)

{

char high[9];

char low[9];

tohex(high, uword >> 8);

tohex(low, uword & 0x00FF);

printf("\nAX: hex(%s%s)", high, low);

tobin(high, uword >> 8);

tobin(low, uword & 0x00FF);

printf("\tbin(%s%s)\n", high, low);

}

void indidication(void)

{

printf("724402-1 Boeshko D.U. LR6 'Funktsii BIOS'\n");

printf("\n1. INT 11h (read list of equipment)\n2. INT 12h(size of continuous memory)\n3. INT 16h(keyboard input output)\n4. INT 1Ah (date and time)\nPress ESC to exit\n");

}

typedef struct \_HDWCFG

{ // bite structury

unsigned HddPresent : 1; //0

unsigned NpuPresent : 1; //1

unsigned AmountOfRAM : 2; //2-3

unsigned VideoMode : 2; //4-5

unsigned NumberOfFdd : 2; //6-7

unsigned DmaPresent : 1; //8

unsigned NumberOfCom : 3; //9-11

unsigned GamePresent : 1; //12

unsigned JrComPresent : 1; //13

unsigned NumberOfLpt : 2; //14-15

} HDWCFG;

void main()

{

long hour,minute,second;

int key, pick;

union REGS rg;

HDWCFG HdwCfg;

unsigned uword;

clrscr();

indidication();

while (key!=27) // poka ne najata klavisha ESC

{

key = getch();

switch (key)

{

case 27:

exit(1);

break;

case 49: // if 1, then

rg.h.ah = 0x0; // Vyzyvaem preryvanie INT 11h dlya polucheniya

int86(0x11, &rg, &rg); //slova konfiguracii komp'yutera

uword = (unsigned int)rg.x.ax; // Poluchaem slovo konfiguracii i sohranyaem

showWord(uword);

memcpy(&HdwCfg, &uword, 2); // v structure HdwCfg

if (HdwCfg.HddPresent)

printf("\nHDD present");

if (HdwCfg.NpuPresent)

printf("\nNPU present");

printf("\nRAM banks: %d", HdwCfg.AmountOfRAM);

printf("\nVideo Mode: %d", HdwCfg.VideoMode);

printf("\nNumver of FDD: %d", HdwCfg.NumberOfFdd + 1);

if (HdwCfg.DmaPresent)

printf("\nDMA present");

printf("\nNumber of COM ports: %d", HdwCfg.NumberOfCom);

if (HdwCfg.GamePresent)

printf("\nGame Adapter present");

if (HdwCfg.JrComPresent)

printf("\nPCjr Com present");

printf("\nNumber of LPT ports: %d", HdwCfg.NumberOfLpt);

printf("\nEnter to continue");

getch();

clrscr();

indidication();

break;

case 50: // if 2 , then

rg.h.ah = 0x0; // Vyzyvaem preryvanie INT 12h dlya opredeleniya

int86(0x12, &rg, &rg); // ob"ema osnovnoj operativnoj pamyati komp'yutera

printf("\nRAM installed: %d Kbytes", (unsigned int)rg.x.ax);

printf("\nEnter to continue");

getch();

clrscr();

indidication();

break;

case 51: // if 3 ,then

printf("\n1.Input a symbol\n2.Input a string\n3.Input with delay\n");

while (pick<49 || pick>51)

{

pick = getch();

}

switch (pick)

{

case 49:

printf("\nInput the symbol:\n");

rg.h.ah = 0; // Vyzyvaem preryvanie INT 16h dlya

int86(0x16, &rg, &rg); // vvoda simvola

printf("\nSymbol is %c", rg.h.al); //i vyvoda ego na ekran

break;

case 50:

printf("\nPRESS ESC TO EXIT\nThe string is:");

while(rg.h.ah != 1)

{

rg.h.ah = 0; // Vyzyvaem preryvanie INT 16h dlya

int86(0x16, &rg, &rg); // vvoda stroki

if(rg.h.ah == 0x1c)

{

printf("\n");

continue;

}

printf("%c", rg.h.al); // i vyvoda ego na ekran

}

break;

case 51:

printf("PRESS ESC TO EXIT\nInput the string:\n");

while (rg.h.ah != 1)

{

rg.h.ah = 0; // Vyzyvaem preryvanie INT 16h dlya

int86(0x16, &rg, &rg); // vvoda stroki

delay(500);

printf("%c", rg.h.al); // i vyvoda ego na ekran

}

break;

case 27:

break;

}

pick = 0;

rg.h.ah = 0;

printf("\nEnter to continue");

getch();

clrscr();

indidication();

break;

case 52:

printf("\n\nCalling the INT 1Ah \n");

printf("\n1.Current date and time\n2.Set date and time\n");

while (pick<49||pick>50)

{

pick = getch();

}

switch (pick)

{

case 49:

rg.h.ah = 0x02; //Schityvanie vremeni s chasov real'nogo vremeni

int86(0x1a, &rg, &rg);

printf("\nTime: %02.2d:%02.2d:%02.2d\n",

bcd1bin(&(rg.h.ch)), // znachenie chasov

bcd1bin(&(rg.h.cl)), // znachenie minut

bcd1bin(&(rg.h.dh)) // znachenie sekund

);

rg.h.ah = 0x04; //Schityvanie vremeni s chasov real'nogo vremeni

int86(0x1a, &rg, &rg);

printf("\nDate: %02d.%02d.%d\n",

bcd1bin(&(rg.h.dl)), // znachenie dnya

bcd1bin(&(rg.h.dh)), //znachenie mesyaca

bcd2bin(&(rg.h.cl)) // znachenie goda

);

break;

case 50:

do

{

printf("\n Enter hour (0-23): ");

scanf("%li", &hour);

} while (hour > 23 || hour<0);

do

{

printf("\n Enter minute (0-59): ");

scanf("%li", &minute);

} while (minute > 59 || minute <0);

do

{

printf("\n Enter second (0-59): ");

scanf("%li", &second);

} while (second > 59 || second<0);

rg.h.ah = 0x03;

bin1bcd(hour, &(rg.h.ch));

bin1bcd(minute, &(rg.h.cl));

bin1bcd(second, &(rg.h.dh));

int86(0x1a, &rg, &rg);

break;

}

pick = 0;

printf("\nEnter to continue");

getch();

clrscr();

indidication();

break;

default:

break;

}

}

}

1. **Скриншоты работы программы**

Ниже представлены скриншоты работы программы (Рисунок 5.1 – 5.6):

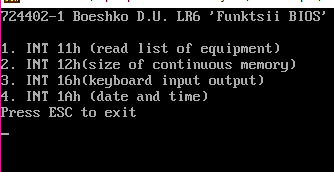


Рисунок 5.1 - Начало работы программы

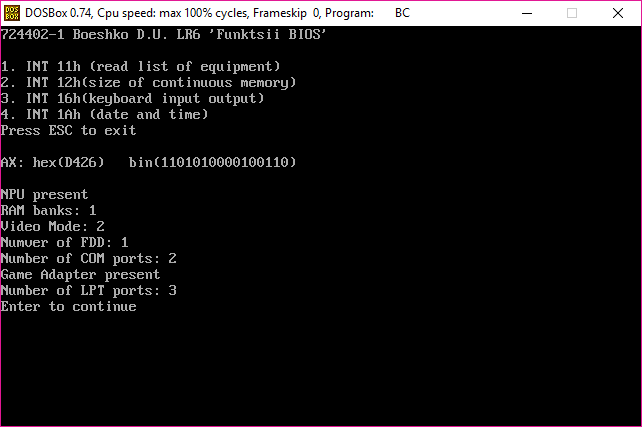


Рисунок 5.2 – Отображение списка оборудования

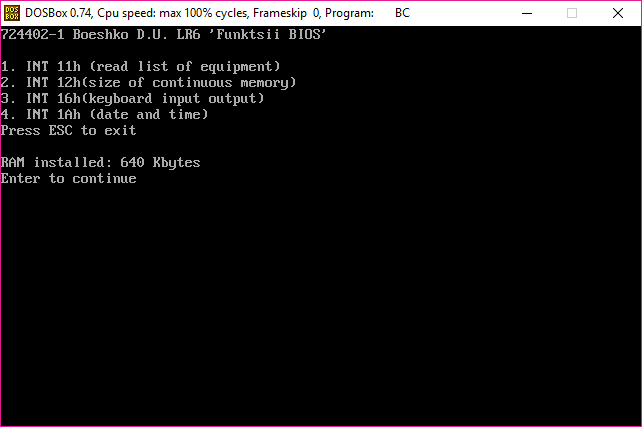


Рисунок 5.3 - Размер непрерывной памяти

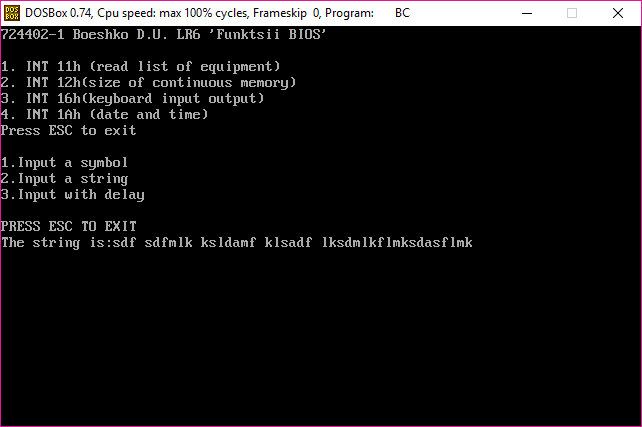


Рисунок 5.4 - Ввод строки

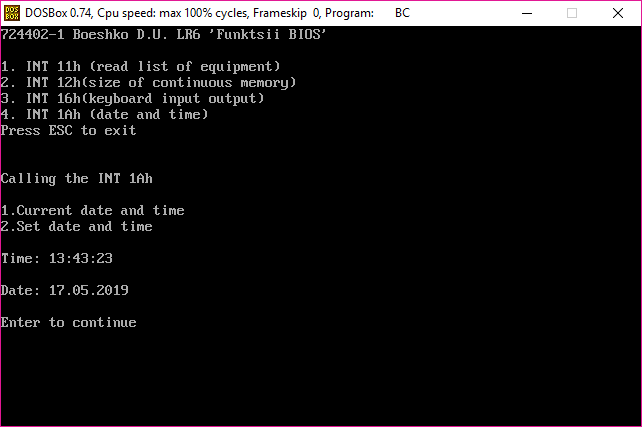


Рисунок 5.5 - Вывод даты и времени

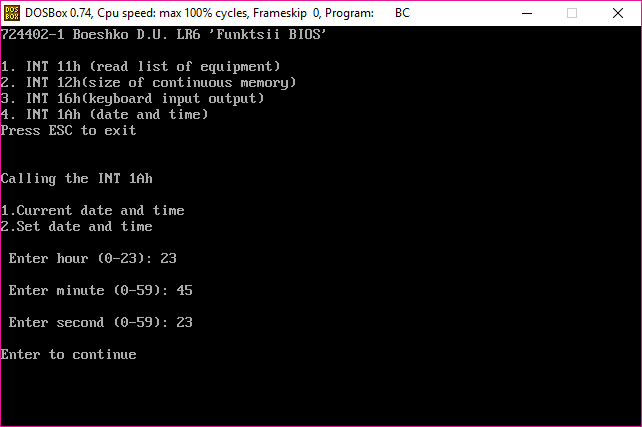


Рисунок 5.6 – Ввод времени

1. **Вывод**

В ходе выполнения лабораторной работы были изучены функции BIOS, использованы векторы прерываний различных назначений, получены данные о компьютере и его составляющих.