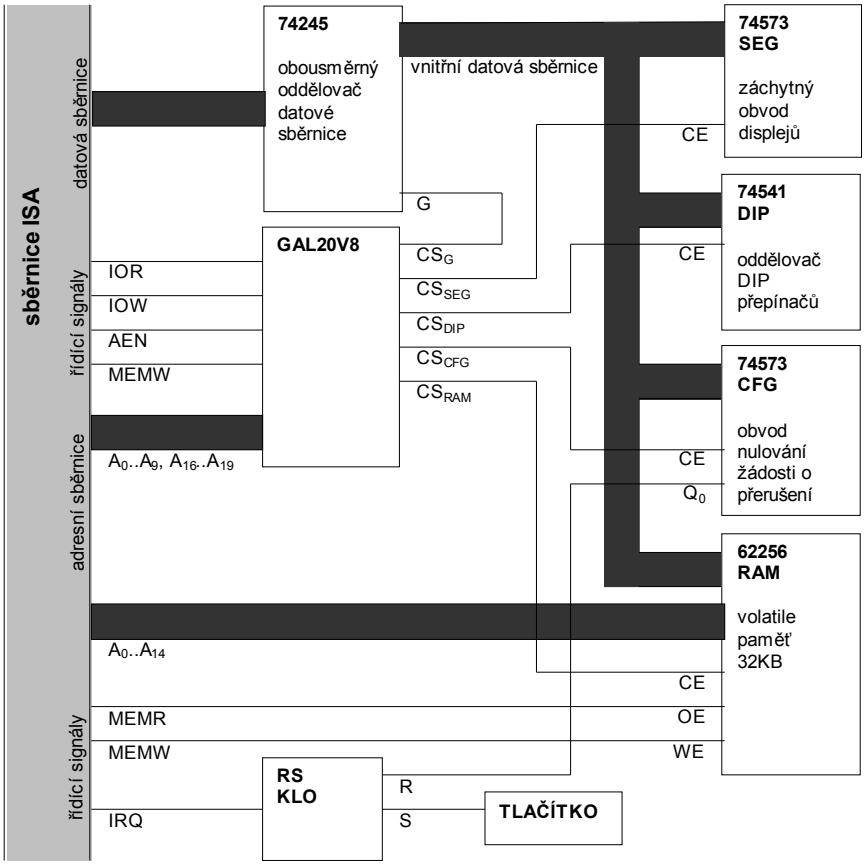
Fakulta informačních technologií VUT v Brně Ústav informatiky a výpočetní techniky	10
Personální počítače, technická péče – cvičení	
# Adapter ROM BIOS	

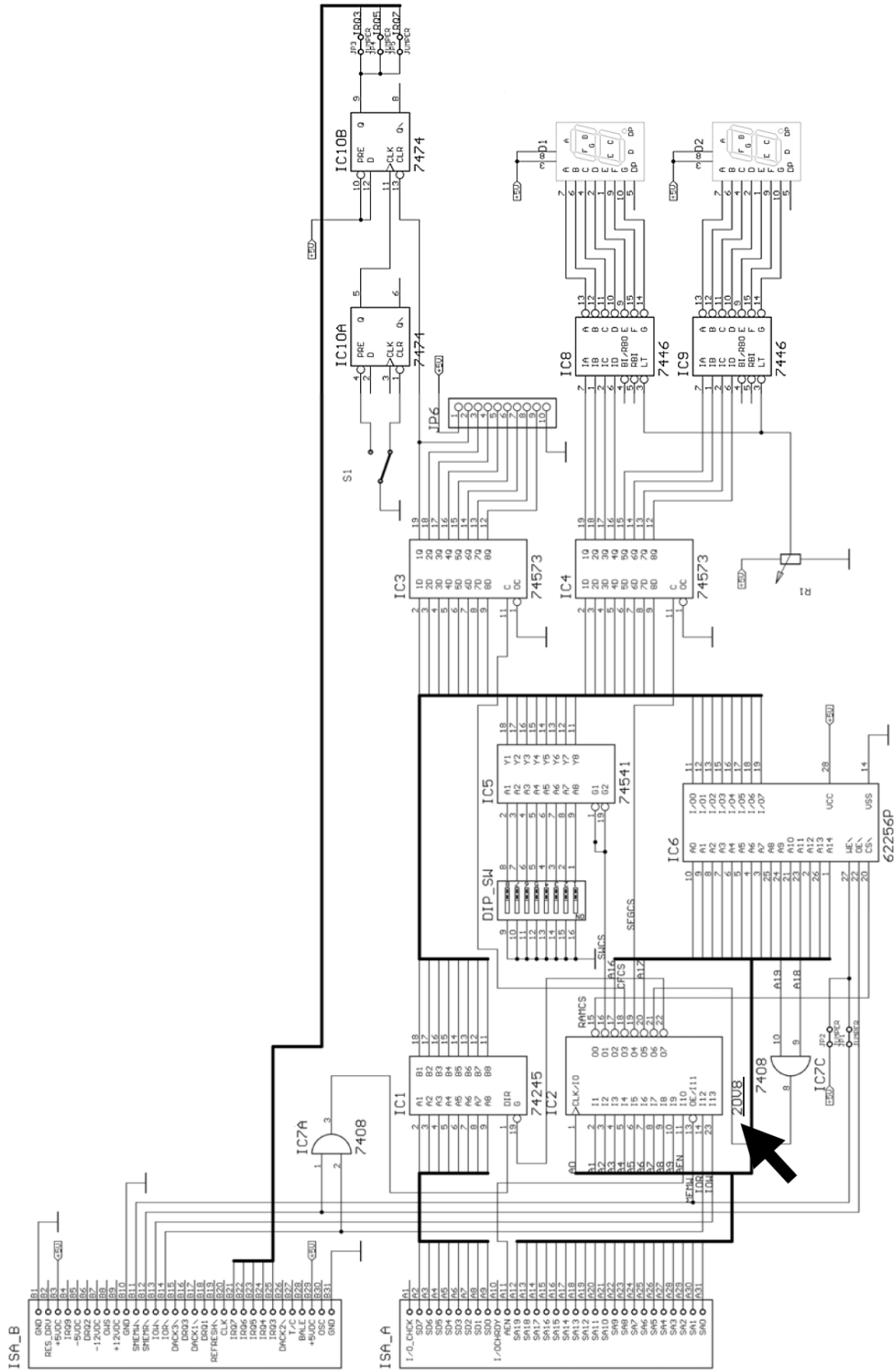
Zadání

1. Seznamte se s konstrukcí cvičné zásuvné adaptérové desky do PC, zejména pak s konstrukcí paměťového podsystému.
2. Seznamte se s mechanismem rozpoznávání přídavných ROM BIOSu při Power On Self Testu.
3. Navrhněte obsah paměti RAM na desce tak, aby bylo po restartu počítače předáno řízení do obslužné rutiny v této paměti. Tato rutina by se měla projevit (vizuálně nebo akusticky) a po stisku klávesy vrátit řízení BIOSu, který by plynule pokračoval v zavádění operačního systému.

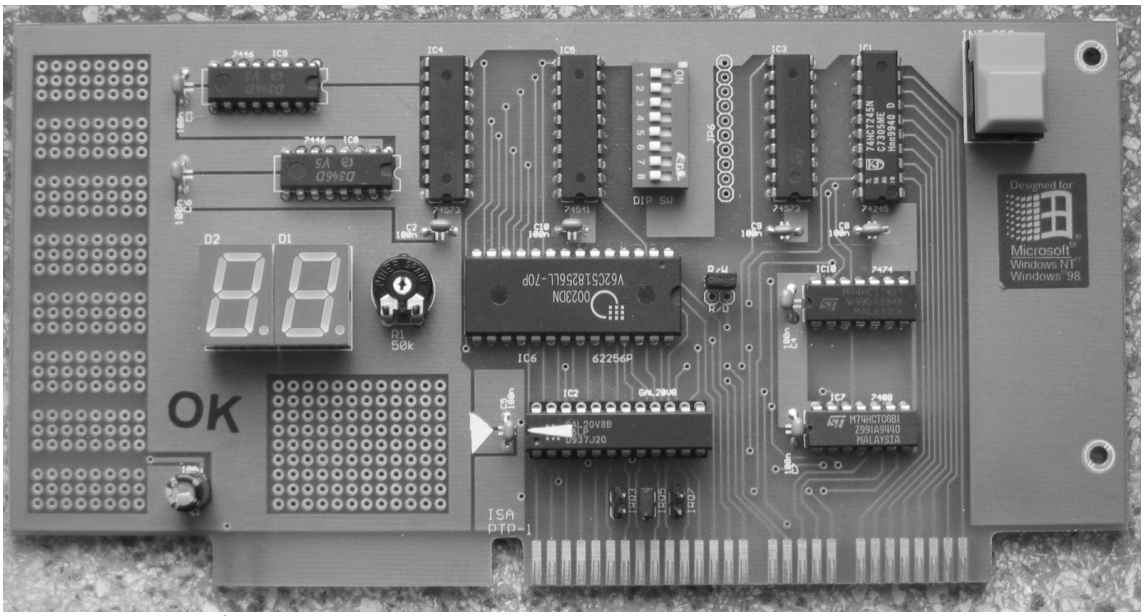
Blokové schéma adaptérové desky



Detailní schéma adaptérové desky



Adaptérová deska



Adaptérová deska ISA[1] obsahuje následující:

- programovatelný adresový dekodér (GAL20V8[2])
- paměť RAM (32KB, 32768×8b, IO 62256)
- vstupní port DIP spínačů
- výstupní port LED segmentového displeje
- výstupní port potvrzení hardwarového přerušení
- tlačítko pro generování hardwarového přerušení

Výstupní port potvrzování hardwarového přerušení je na adrese **`300h`**, výstupní port LED segmentového displeje má adresu **`301h`**, vstupní port DIP spínačů má adresu **`302h`**. Paměť RAM je v segmentu **`D000h`**.

Mechanizmus vyhledávání ROM:

BIOS počítače hledá tzv. adapter ROMs od adresy 0C800:0000h po 2KB blocích. Pokud je na začátku bloku nalezena hlavička podle tab. 1, je kontrolován odpovídající blok paměti mechanismem ověření jednoduchého kontrolního součtu. Délka bloku je rovněž zjištěna z hlavičky. Mechanismus kontrolního součtu sečte všechny bajty udaného bloku modulo 256. Součet musí být nulový. To se obyčejně zajišťuje posledním bajtem obsahu ROM, který nemá žádný význam kromě toho, že je dvojkovým doplňkem součtu užitečného obsahu ROM. Zbytek bajtů do nejbližší adresy dělitelné 2048 bývá pro zajištění jejich neutrality při kontrole sečítáním nulový. Celý mechanismus vyhledávání adapter ROMs je vidět ve výpisu úseku zdrojového kódu hlavního BIOSu IBM PC AT na příloze 1 (převzato z [1]).

[1] ISA = Industry Standard Architecture, 16bitová sběrnice pro PC AT, navržená v roce 1984
[2] programovatelné logické pole, EECMOS

Offset	*Obsah*
0	55h – Identifikace hlavičky
1	AAh – Identifikace hlavičky
2	Délka ROM v 512B blocích
3	Proveditelný kód
…	…
n	Dvojkový doplněk ke kontrolnímu součtu

Tabulka 1Struktura paměti ROM

První dva bajty musí nutně být 55h a AAh, neboť se podle nich poznává přítomnost ROM. Další bajt je délka ROM v 512B blocích. Například ROM s délkou 32kB bude mít zde číslo 64 (40h). Za tímto bajtem může následovat spustitelný kód. Návrat řízení BIOSu na systémové desce musí být realizováno instrukcí RETF.

Celý obsah ROM (vč. hlavičky) je zajištěn kontrolním součtem, který je dán součtem všech bajtů obsažených v ROM mod 256. Takto spočítaný kontrolní součet musí mít nulovou hodnotu.

Postup práce:

1. Nastudujte mechanizmus poznávání ROM BIOS při Power On Self Testu. Soustřeďte se na formát hlavičky a mechanizmus počítání kontrolního součtu.
2. Nakopírujte si soubor `C:\PTP-ITP\sablona.zip` do Vašeho adresáře a rozbalte. Po rozbaleni budete mit k dispozici soubory `example.asm`, `template.asm` a `compile.bat`, které mají následující význam:
   - `example.asm` – ukázkový příklad v assembleru, který zapne PC speaker a vyplní obrazovku znaky A. Program má stejnou funkci jako kód, který bude nahrán do paměti na adaptéru a bude aktivován při startu počítače.
   - `template.asm` – šablona programu, který musí zajistit správné nahrání rutiny BOOT_ROM do paměti na adaptéru tak, aby byla rutina vyvolaná při startu počítače (při Power On Self Test). Program obsahuje prázdná místa, do kterých je nutné doplnit správný kód.
   - `compile.bat` – dávka, pomocí které je možné přeložit soubory z assembleru do spustitelného tvaru.
3. Pomocí programu `example.asm` si můžete vyzkoušet, jak se bude projevovat obslužná rutina zapsaná programem `template.asm` do paměti adaptéru ISA. Program si můžete přeložit pomocí dávky `compile.bat`. Po spuštění by se měl rozeznít PC speaker a obrazovka se vyplnit znakem A. Stiskem klávesy ESC se PC speaker vypne a program se ukončí.
4. Upravte program `template.asm` tak, aby rutina uložená za návěštím BOOT_ROM byla nahrána do paměti adaptéru na adresu 0D000h a spuštěna při startu PC. K tomuto je nutná následujicí posloupnost kroků:
   a) Přidat na začátek rutiny BOOT_ROM správnou hlavičku.
   b) Na konec rutiny BOOT_ROM přidat správné ukončení.
   c) Je nunté k součtu všech bajtů rutiny BOOT_ROM vytvořit dvojkový doplněk.
   d) Spočtený dvojkový doplněk je nutné zapsat za rutinu BOOT_ROM (návěští chck).

e) Zapsat celou rutinu BOOT_ROM do paměti na adaptéru. Nezapomeňte zkopírovat všech 512 bajtů – jeden celý blok. Pro zapsání programu můžete použít v cyklu instrukci movsb, která kopíruje jeden bajt z adresy DS:[SI] na adresu ES:[DI] a následně inkrementuje registry SI a DI o jedničku. Cyklus je možné implementovat instrukcí rep, která opakovaně provádí bezprostředně navazující instrukci. Například `rep movsb` provede opakovaně instrukci movsb. Hodnota registru CX určuje počet opakování této instrukce.

5. Po úpravě, přeložení a spuštění programu `template.asm` proveďte restart počítače. Při startu by se měla vyvolat rutina nahraná do paměti adaptéru. Chování rutiny by mělo být totožné s programem `example.asm` (zapne se reproduktor, obrazovka se vyplní znakem A a čeká se na stisk klávesy ESC).
6. Dle vlastního uvážení modifikujte obslužnou rutinu a sledujte provedené změny (např. změna vypisovaných znaků, výpis řetězce na obrazovku, atd.)

Použitá literatura

[1] IBM PC AT Technical Reference Manual, IBM 1985

[2] Vrátil, Z.: Architektura PC XT+AT, díl 2., Gethon Audio and Computer, 1992

Příloha 1

Úsek zdrojového kódu hlavního BIOSu IBM PC AT pro vyhledávání adapter ROMs

```
;CHECK FOR OPTIONAL ROM FROM C800->EOOO IN 2K BLOCKS (A VALID MODULE HAS "55AA" IN THE FIRST 2 ;
LOCATIONS LENGTH lNDICATOR (LENGTH/512) IN THE 3RD LOCATION AND TEST/INIT. CODE STARTING IN THE
4TH ;LDCATION)
ROM_SCAN1:
       STI                  ;ALLOW INTERRUPTS
       MOV     AL,3BH               ;<><><><><><><><><><><><><>
       OUT     MFG PORT,AL    ;<><> CHECKPOINT 3B  <><>
       CALL    DDS            ;SET REAL MODE DATA SEGMENT
       MOV     AL,1O          ;LINE FEED ON DISPLAY
       CALL    PRT_HEX

ROM_SCAN:
       ; --- SET DMA MASK AND REQUEST REGISTERS

       SUB     AL,AL
       OUT     DMA18+2,AL     ;SEND ZERO TO MASK REGISTER
       JMP     $+2
       OUT     DMA18+4,AL     ;SEND ZERO TO REQUEST REGISTER
       MOV     DX,0C800H      ;SET BEGINNING ADDRESS
ROM_SCAN2:
       MOV     DS,DX
       PUSH    DI             ;SAVE WORK REGISTER
       MOV     DI,0AA55H      ;GET TEST PATTERN
       SUB     BX,BX          ;SET BX=OOOO
       MOV     AX,[BX]               ;GET 1ST WORD FROM MODULE
       CMP     AX,DI          ;= TO ID WORD?
       POP     DI             ;RECOVER WORK REGISTER
       JNZ     NEXT_ROM       ;PROCEED TO NEXT ROM IF NOT
       CALL    ROM_CHECK      ;GO CHECK OUT MODULE
       JMP     SHORT ARE_WE_DONE ;CHECK FOR END OF ROM SPACE
NEXT_ROM:
       ADD     DX,0080H       ;POINT TO NEXT 2K ADDRESS
ARE_WE_DONE:
       CMP     DX,0E000H      ;AT E0000 YET?
       JL      ROM_SCAN2      ;GO CHECK ANOTHER ADD. IF NOT
.                             ;TEST FOR KEYBOARD LOCKED
.
.
;ROM CHECKSUM SUBROUTlNE

ROM_CHECKSUM   PROC    NEAR
       SUB     CX,CX          ;NUMBER OF BYTES TO ADD IS 64K

ROM_CHECKSUM_CNT:             ;ENTRY FOR OPTIONAL ROM TEST
       XOR     AL,AL
ROM_L:
       ADD     AL,[BX]               ;GET [DS,BX]
       INC     BX             ;POINT TO NEXT BYTE
       LOOP    ROM_L          ;ADD ALL BYTES IN ROM MODULE

       OR      AL,AL          ;SUM = 0?
       RET

ROM_CHECKSUM   ENDP

;THIS ROUTlNE CHECKSUMS OPTIONAL ROM MODULES AND IF CHECKSUM IS OK, CALLS INITIALIZATION/TEST
CODE ;IN MODULE

ROM_CHECK      PROC    NEAR
       MOV     AX,DATA               ;POINT ES TO DATA AREA
       MOV     ES,AX
       SUB     AH,AH          ;ZERO OUT AH
       MOV     AL,[BX+2]      ;GET LENGTH INDICATOR
       SHL     AX,9           ;MULTIPLY BY 512
       MOV     CX,AX          ;SET COUNT
       SHR     AX,4
```

```
        ADD     DX,AX           ;SET POINTER TO NEXT MODULE
        CALL    ROM_CHECKSUM_CNT;DO CHECKSUM
        JZ      ROM_CHECK_1

        CALL    ROM_ERR                 ;POST CHECKSUM ERROR
        JMP     SHORT ROM_CHECK_END ;AND EXIT

ROM_CHECK_1:
        PUSH    DX              ;SAVE POINTER
        MOV     ES:@IO_ROM_INIT,0003H ;LOAD OFFSET
        MOV     ES:@IO_ROM_SEG,DS ;LOAD SEGMENT
        CALL    DWORD PTR ES:@IO_ROM_INIT  ;CALL INITIALIZE/TEST ROUTlNE
        POP     DX

ROM_CHECK_END:
        RET                     ;RETURN TO CALLER

ROM_CHECK       ENDP
```