SDC_One – komputer zdefiniowany programowo z klasycznym mikroprocesorem (7)

Retrokomputer

Kolejna część cyklu prezentuje możliwości użycia komputera SDC_One do pracy z oprogramowaniem z epoki komputerów domowych. Ten nieco bardziej zabawowy sposób użycia komputera daje możliwość przeniesienia się w czasie i przyjrzenia się na żywo historii informatyki.

W sieci można łatwo znaleźć oprogramowania dla 8-bitowych komputerów z lat 1970-tych i 1980-tych. Część oprogramowania dostępna jest w postaci źródłowej, co umożliwia łatwe dostosowanie go do konkretnego modelu komputera

Interpreter języka BASIC dla 6502

W Internecie można znaleźć asemblerowy kod źródłowy interpretera języka BASIC, napisanego w roku 1977 – w początkach rozwoju firmy Microsoft, a używanego w wielu komputerach domowych, w tym maszynach Apple, Atari i Commodore 64. Interpreter ten zajmuje ok. 8 KiB pamięci i wykonuje obliczenia na liczbach stało- i zmiennopozycyjnych. Prawdopodobnie pierwszą platformą, na której był on używany, był komputer firmy Ohio Scientific. Interpreter jest znany pod nazwami m.in. OSI Basic, Applesoft i Commodore BASIC. Dostosowanie interpretera do nowej platformy wymaga jedynie napisania własnych procedur obsługi terminala.

Język BASIC był jednym z pierwszych języków programowania komputerów. Interpreter pełnił równocześnie rolę edytora i prostego systemu operacyjnego. Jako ciekawostkę można współcześnie traktować sposób wprowadzania i edycji programów. Interpreter pracuje w trybie wierszowym. Jeżeli wpisany wiersz rozpoczyna się od liczby, jego zawartość jest rozpoznawana i zapisywana do pamięci jako fragment programu. W przeciwnym razie interpreter natychmiast wykonuje wprowadzone polecenie. Polecenie RUN powoduje wykonanie zapamiętanego programu. Kolejność instrukcji odpowiada wprowadzonym przez użytkownika numerom linii programu.

Programy w języku BASIC mogą być pisane na terminalu komputera. Można je również przygotowywać na komputerze PC i przesyłać do terminala komputera poprzez przeciągnięcie pliku źródłowego do okna terminala.

Dla SDC_One z procesorem W65C02S zaadaptowano dostępną w sieci wersję OSI Basic pochodzącą ze strony ***http://bit.ly/2MS69Pv***. Jej kod źródłowy może być asemblowany dostępnym w sieci asemblerem ca65, wchodzącym w skład pakietu cc65. Adaptacja programu wymagała napisania procedur współpracy z terminalem, drobnej modyfikacji procedury skanowania pamięci RAM, tak, by nie próbowała ona podczas określania dostępnego rozmiaru dostępnej pamięci RAM zamazać umieszczonego również w pamięci RAM interpretera oraz korekt skryptu konsolidatora (**listing 1…4**).

Listing 1. Procedury obsługi terminala dla interpretera OSI BASIC (plik OSI_BAS.S)

```
; STARTUP AND SERIAL I/O ROUTINES ======
; SDC_One port by gbm
MINIIO_BASE := $FE00
IO_CON      := MINIIO_BASE + 1
IO_CSR      := MINIIO_BASE + 3
IO_CSR_RXNE := 1
IO_CSR_TXE  := 2

.segment "IOHANDLER"
.org $FF00
Reset:
    LDX     #STACK_TOP
    TXS

; Display startup message
    LDY #0
ShowStartMsg:
    LDA  StartupMessage,Y
    BEQ  WaitForKeypress
    JSR  MONCOUT
    INY
    BNE  ShowStartMsg

; Wait for a cold/warm start selection
WaitForKeypress:
    JSR  MONRDKEY
    BCC  WaitForKeypress
    AND  #$DF    ; Make upper case
    CMP  #'W'    ; compare with [W]arm start
    BEQ  WarmStart
    CMP  #'C'    ; compare with [C]old start
    BNE  Reset
    JMP  COLD_START  ; BASIC cold start

WarmStart:
    JMP  RESTART  ; BASIC warm start

MONCOUT:
    PHA
SerialOutWait:
    LDA  IO_CSR
    AND  #IO_CSR_TXE
    BEQ  SerialOutWait  ; opt gbm
    PLA
    STA  IO_CON
    RTS

MONRDKEY:
    LDA  IO_CSR
    AND  #IO_CSR_RXNE
    CMP  #1
    BNE  NoDataIn   ; opt gbm
    LDA  IO_CON
    SEC    ; Carry set if key available
    RTS
NoDataIn:
    CLC    ; Carry clear if no key pressed
    RTS
```

Oprócz pakietu CC65 do przygotowania programu potrzebny jest również program do konwersji plików .HEX. Do tego celu użyto programu SREC_CAT, wchodzącego w skład pakietu SRECORD. Ponieważ SDC_One udostępnia 64 KiB pamięci RAM, interpreter został skonfigurowany do umieszczenia go w pamięci począwszy od adresu 0xD000. Przykładową sesję z interpreterem BASIC pokazano na **rysunku 1**.

Listing 2. Modyfikacja testu rozmiaru pamięci (plik OSI_BAS.S)

```
L40D7:
    inc     LINNUM
    bne     L40DD
    inc     LINNUM+1
; SDC_One - stop memory test begore interpreter area
    Lda  LINNUM+1
    Cmp  #>BASIC_START
    Beq  MEMSIZE_DONE
L40DD:
```

Uruchomienie systemu operacyjnego CP/M-80

Po uruchomieniu SDC_One pojawił się pomysł zainstalowania na nim systemu operacyjnego CP/M-80, używanego w komputerach z mikroprocesorami 8080, 8085 i Z80, wyposażonymi w jednostki pamięci masowej – napędy dyskietek. W sieci można łatwo znaleźć system CP/M i jego dokumentację, w tym dokładny opis instalacji systemu na dowolnym komputerze. System CP/M-80 może być obecnie instalowany i używany bez opłat do celów niekomercyjnych.

Listing 3. Skrypt konsolidatora ld65 dla interpretera OSI BASIC

```
MEMORY {
    ZP:  start = $0000, size = $0100, type = rw;
    BASROM:  start = $D000, size = $2F00, fill = yes, fillval=$FF, file = %O;
    IOHANDLER:  start = $FF00, size = $FA, fill = yes, fillval=$FF, file = %O;
    VECTS:  start = $FFFA, size = $6, fill = yes, fillval=$FF, file = %O;
}

SEGMENTS {
    ZEROPAGE:  load = ZP, type = zp;
    CODE:  load = BASROM, type = ro;
    IOHANDLER:  load = IOHANDLER, type = ro;
    VECTS:  load = VECTS, type = ro;
}
```

Listing 4. Plik wsadowy assemble.bat

```
ca65 osi_bas.s -o osi_bas.o -l
ld65 -C osi_bas.cfg osi_bas.o -o osi_bas.bin
srec_cat osi_bas.bin -binary -offset 0xd000 -unfill 0xff 16 -o osi_bas.hex -intel
pause
```

Zrekonstruowana postać źródłowa bazowej części systemu (CCP+BDOS) jest dostępna w sieci w dwóch wersjach, zapisanych w językach asemblerowych 8080 oraz Z80. Część ta jest niezależna od sprzętu, dlatego najłatwiej posłużyć się wersją ładowalną w postaci binarnej lub Intel .HEX. Jeżeli dysponujemy wersją binarną – można ją przetworzyć na postać .HEX, ładowaną przez monitor sprzętu. Należy jedynie pamiętać, że ma ona być załadowana pod adres 0xE400.

Uwaga! Ze względu na znaczną objętość listing 5 został umieszczony w materiałach dodatkowych do artykułu i może być pobrany z serwera ftp.

Dostosowanie systemu do komputera polega na napisaniu następujących modułów asemblerowych:

- CBIOS.ASM – zawierającego procedury specyficzne dla sprzętu, niezbędne do działania systemu CP/M. Moduł tej jest częścią systemu CP/M.
- GETSYS.ASM – program ładujący systemu CP/M, umieszczony w pierwszym sektorze nośnika pamięci masowej,
- PUTSYS.ASM – program zapisujący obraz systemu CP/M z pamięci na nośniku pamięci masowej.
- WRBOOT.ASM – program zapisujący postać binarną GETSYS do pierwszego sektora nośnika pamięci masowej.

Do asemblacji można użyć asemblera TASM (Table-driven assembler).

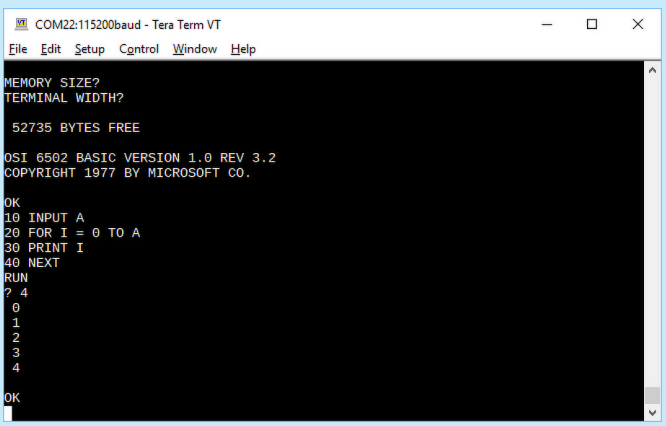
Wersja modułu CBIOS.ASM dla SDC_One liczy ok. 300 wierszy kodu asemblerowego (**listing 5** – dostępny w materiałach dodatkowych). Została ona napisana w asemblerze 8080, dzięki czemu może być użyta w wersjach SDC_One wyposażonych w procesory 8085 i Z80.

Moduł CBIOS.ASM zawiera kilka istotnych procedur oraz tablice z parametrami jednostek pamięci masowej. System CMP operuje na jednostkach dyskowych z sektorami o rozmiarze 128 B.W oprogramowaniu SDC_One zrealizowano dwie jednostki pamięci masowej. Pierwsza z nich ma pojemność 512 KiB i składa się z 256 ścieżek zawierających po 16 sektorów Jej zawartość jest przechowywana w pamięci Flash mikrokontrolera i buforowana w dwóch buforach ścieżek, umieszczonych w pamięci RAM. Rozmiar ścieżki odpowiada rozmiarowi strony pamięci Flash mikrokontrolerów serii STM32L47x. Ponieważ pierwsza przechowuje system operacyjny, parametr określający liczbę zarezerwowanych ścieżek ma wartość 4 (system zajmuje 7 KiB i mieści się w czterech ścieżkach po 2 KiB). Druga jednostka ma 16 ścieżek o tym samym rozmiarze. Jej zawartość jest przechowywana w bloku pamięci RAM2 mikrokontrolera STM32L476. Może być ona zapisana w pamięci Flash przy użyciu polecenia monitora. Przy starcie komputera jej zawartość jest odtwarzana z pamięci Flash mikrokontrolera. Pierwsza jednostka służy głównie do przechowywania plików o stałej zawartości – systemu operacyjnego i programów użytkowych. Druga została przewidziana jako dysk roboczy, służący do przygotowywania programów. Moduł CBIOS jest umieszczony na końcu pamięci zajmowanej przez system CP/M. Może on mieć rozmiar do 1,5 kB.

Program PUTSYS.ASM (**listing 6**) służy do zapisania obrazu systemu z pamięci RAM komputera do pamięci masowej. Został on napisany w taki sposób, by mógł być załadowany pod adres 0x100. Pierwszy etap przygotowania systemu polega na asemblacji programów i stworzeniu trzech plików ładowalnych .HEX:

- CPM22.HEX – części systemu niezależnej od sprzętu,
- CBIOS.HEX – części systemu zależnej od sprzętu,
- PUTSYS.HEX – programu zapisującego system.

Te trzy pliki należy załadować do pamięci SDC_One, korzystając z monitora sprzętu. Następnie pod adres 0 należy zapisać instrukcję skoku do adresu 0x100, która posłuży do uruchomienia programu PUTSYS. Po zresetowaniu procesora i uruchomieniu go poleceniem run. Program PUTSYS zostanie wykonany i system zostanie zapisany do pamięci masowej.



Rysunek 1. Sesja z interpreterem BASIC – program przykładowy

Drugi etap przygotowania systemu – to zapis programu ładującego. Program ładujący GETSYS.ASM jest bardzo podobny do programu PUTSYS.ASM. Jedyne różnice – to użycie polecenia odczytu sektora pamięci masowej zamiast zapisu oraz skok do systemu po załadowaniu go. Podobnie jak PUTSYS, moduł GETSYS wykonuje się spod adresu 0x100.

Program GETSYS jest umieszczony w pierwszym sektorze pierwszej jednostki pamięci masowej. Ładuje on system CP/M-80 pod adresy położone na końcu pamięci RAM. Ponieważ cały CP/M 80 zajmuje 7 kB, rozpoczyna się on od adresu 0xE400 (**listing 7**). Do zapisu programu ładującego GETSYS potrzebny jest program zapisujący pierwszy sektor nośnika. Nosi on nazwę WRBOOT.ASM (**listing 8**). Jest on ładowany pod adres 0 i uruchamiany po zresetowaniu procesora. W celu zapisania programu ładującego należy załadować do pamięci komputera pliki .HEX powstałe przez translację modułów PUTSYS i WRBOOT, a następnie zresetować procesor i uruchomić program WRBOOT, umieszczony pod adresem 0.

Ostatnim potrzebnym fragmentem oprogramowania jest program BOOT.ASM (**listing 9**), ładujący pierwszy sektor nośnika pamięci masowej do pamięci RAM komputera pod adres 0x100 i przekazujący sterowania do załadowanego programu ładującego. Jest on podobny (symetryczny) do WRBOOT.ASM. Program ten musi być umieszczony w pamięci komputera. W rzeczywistych komputerach rezyduje on w pamięci nieulotnej. W przypadku SDC_One jest on automatycznie wpisywany do pamięci komputera przez monitor sprzętu przy starcie komputera.

Listing 6. PUTSYS.ASM dla SDC_One

```
; Program writing memory image of CP/M system to disk.
; image from 0xE400 is copied to the disk, starting with 2nd sector
; - the 1st sector is used by bootloader
; This program is loaded to the TPA (address 0x100)
; Julia Kosowska 01'2017

#include "miniio.h"

ccp  .EQU  0e400h  ;base address of CP/M
mem  .EQU  010000h ;base address of CP/M
size  .EQU mem-ccp
nsects  .EQU       (size+127)/128 ;number of sectors to load
SPT .EQU  16
    .ORG  100H
    xra  a
    OUT  IO_MS_UNIT
    OUT  IO_MS_TRK
    Mvi  B,nsects
    mvi  C,1
    lxi  H,ccp
    lxi  D,128
next_sec:
    mov  A,C
    OUT  IO_MS_SEC
    mov  A,L
    OUT  IO_MS_A0
    mov  A, H
    OUT  IO_MS_A1
    mvi  a,IO_MS_CMD_WR
    out  IO_MS_CMD  ; write command
    DAD  D  ; next sector adress
    INR  C
    DCR  B
lop:
    JZ   fin
    mov  A,C
    CPI  SPT
    JNZ  next_sec
; advance to next track
    IN  IO_MS_TRK
    INR  A
    OUT  IO_MS_TRK
    mvi  C,0
    jmp  next_sec

fin:  ; flush buffer
    mvi   a, IO_MS_CMD_SYNC
    out   IO_MS_CMD ; write command
    jmp   $
    .end
```

Listing 7. GETSYS.ASM dla SDC_One

```
; Second stage bootloader program - read memory image of CP/M system from
; disk and jumps to BIOS cold boot routine. This program should be put into
; the first sector of the first track of the default system disk. First stage
; bootloader residing in ROM memory should load this program from disk
; to address 100h and jump to it.
; JK & gbm 01'2017
; assemble with: tasm -85 getsys.asm

#include "miniio.h"

ccp    .EQU         0e400h    ;base address of CP/M
bios   .equ         0FA00h    ;base of bios
mem    .EQU         010000h   ;end address of CP/M
size   .EQU         mem-ccp
nsects  .EQU  (size+127)/128  ;number of sectors to load
SPT  .EQU  16

    .ORG  100H
    xra  a
    OUT  IO_MS_UNIT
    OUT  IO_MS_TRK
    mvi  B, nsects
    mvi  C, 1   ; start with trk 0 sector 1
    lxi  H, ccp ; load address
    lxi  D, 128 ; sector size
next_sec:  ; load a sector
    mov  A, C
    OUT  IO_MS_SEC
    mov  A, L
    OUT  IO_MS_A0
    mov  A, H
    OUT  IO_MS_A1
    mvi  a, IO_MS_CMD_RD
    out  IO_MS_CMD  ; read
    DAD  D     ; next sector adress
    INR  C
    DCR  B
    JZ   bios  ; start the OS
    mov  A, C
    CPI  SPT
    JNZ  next_sec
            ; advance to next track
    IN   IO_MS_TRK
    INR  A
    OUT  IO_MS_TRK
    mvi  C, 0
    jmp  next_sec
    .end
```

Listing 8. WRBOOT.ASM dla SDC_One

```
; Write 128B of RAM memory starting at address 0x100 to the first
; sector of the first track of disk 0.
; JK & gbm 2017
; assemble with: tasm -85 wrboot.asm
#include "miniio.h"
    xra  A
    out  IO_MS_UNIT
    out  IO_MS_TRK
    out  IO_MS_SEC
    out  IO_MS_A0
    inr  A
    out  IO_MS_A1
    mvi  a, IO_MS_CMD_WR
    out  IO_MS_CMD  ; write
    mvi a, IO_MS_CMD_SYNC
    out IO_MS_CMD  ; synchronize
    jmp $
    .END
```

Kopiowanie plików z PC do pamięci masowej komputera

Wynikiem opisanych powyżej czynności jest działający system CP/M-80. Do pracy z systemem są jeszcze potrzebne podstawowe programy użytkowe, które muszą znaleźć się w pamięci masowej naszego komputera. Najprostszym sposobem na to jest użycie wbudowanego polecenia SAVE systemu CP/M.

Plik, który chcemy zapisać w pamięci masowej komputera, zamieniamy na postać Intel .HEX przy użyciu dowolnego programu służącego do tego celu, np. wspomnianego wcześniej SREC_CAT. Jeśli chcemy w ten sposób załadować program i dysponujemy jego postacią źródłową, wystarczy ją zasemblować do postaci .HEX. Postać .HEX musi mieć adres początkowy 0x100, tak, by plik był ładowany do obszaru TPA systemu CP/M, służącego do uruchamiania programów Czynność tę możemy wykonać dla każdego pliku, niezależnie od jego zawartości. W ten sposób możemy przesłać do naszego komputera zarówno programy, jak i pliki tekstowe.

Mając uruchomiony na komputerze system CP/M, ładujemy plik .HEX przy użyciu konsoli monitora sprzętu. Dla przyspieszenie ładowania warto podczas tej operacji zatrzymać procesor. Po załadowaniu pliku .HEX włączamy

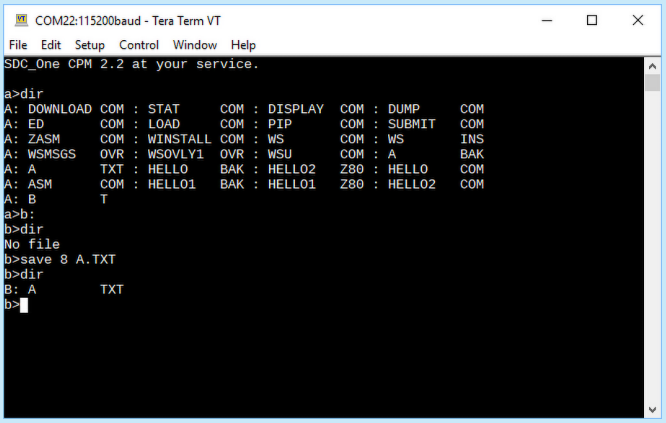
Listing 9. BOOT.ASM dla SDC_One

```
; CP/M-80 bootloader to be placed at address 0
; Read 128B of RAM memory starting at address 0x100 from the first
; sector of the first track of disk 0.
; JK & gbm 2017
; assemble with: tasm -85 boot.asm
#include "miniio.h"
    XRA  A
    OUT  IO_MS_UNIT
    OUT  IO_MS_TRK
    OUT  IO_MS_SEC
    OUT  IO_MS_A0  ; address LSB
    INR  A
    OUT  IO_MS_A1  ; address MSB
    MVI  A, IO_MS_CMD_RD
    OUT  IO_MS_CMD  ; read
    JMP  LOADER
    .org 100h
LOADER:
    .END
```

normalną pracę poleceniem run monitora i w konsoli systemu CP/M wpisujemy polecenie: **save n NAZWA.XXX**, w którym n jest rozmiarem zapamiętywanego pliku w blokach 256-bajtowych wyrażonym w postaci liczby dziesiętnej. Rozszerzenie nazwy pliku musi odpowiadać jego zawartości – dla programów systemu CP/M-80 będzie to .COM.

Zrzut ekranowy podczas pracy systemu CP/M-80 uruchomionego na komputerze SDC_One pokazano na **rysunku 2**.

Julia Kosowska
Grzegorz Mazur



Rysunek 2. System operacyjny CP/M-80 działający na komputerze SDC_One

Pożyteczne adresy

Podczas przygotowania systemu CP/M-80 użyto programów narzędziowych, kodów źródłowych i instrukcji pochodzących z witryn:

- *http://bit.ly/2wj6K29*
- *http://bit.ly/2BFubsy*
- *http://bit.ly/2Ln3f0a*