Marcin Stępniak

# Architektura systemów komputerowych
# Laboratorium 6
# Symulator procesora 8086

## 1. Informacje

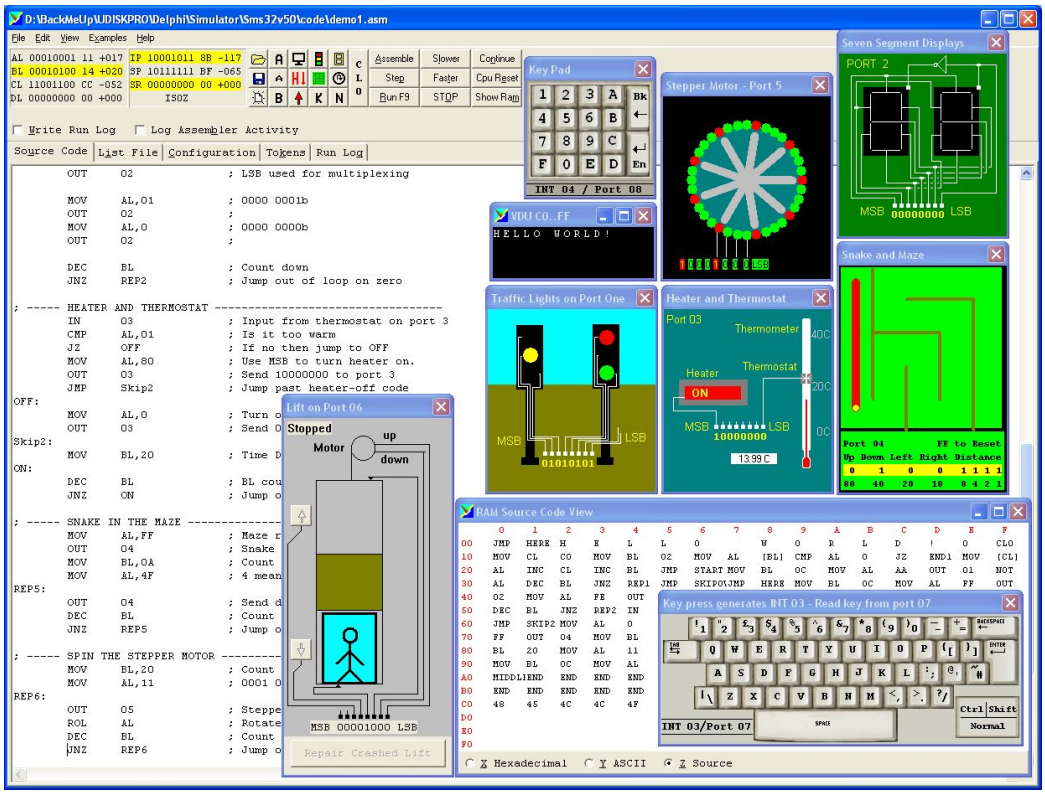
### 1.1. Symulator SMS32

SMS32 jest symulatorem mikroprocesora, który używa zbioru instrukcji zbliżonego do układów Intel 8086. Język maszyny został wymyślony i nie istnieje sprzęt, który pozwala na jego uruchomienie. Powodem stworzenia programu były problemy przy uruchamianiu rzeczywistego kodu w asemblerze. Najdrobniejsze błędy w kodzie powodowały wyłączenie napisanego programu, a czasem nawet zawieszenie lub problemy w systemie operacyjnym.

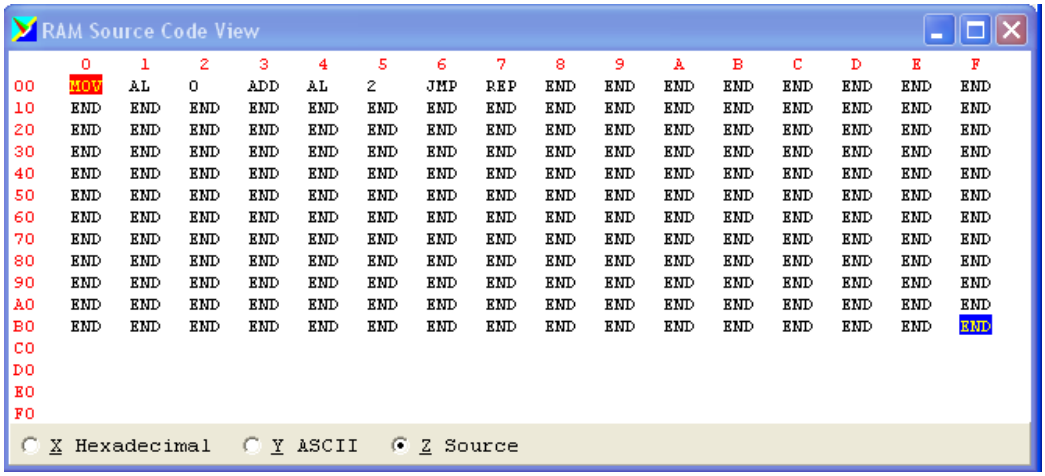
SMS32 pozwala na:

1. poznanie zasad działania mikroprocesorów używających uproszczonej maszyny 8-bitowej,
2. pisanie programów w asemblerze z wykorzystaniem instrukcji podobnych do tych wykorzystywanych w układach rodziny Intel 8086,
3. uruchamianie (w trybie ciągłym lub krok po kroku) programów i obserwowanie zmian rejestrów CPU i pamięci RAM,
4. programowanie symulowanych wejściowych i wyjściowych urządzeń peryferyjnych,
5. poznanie zasad wykorzystania odpytywania (ang. polling) i przerwań.

Symulator można pobrać pod adresem: http://www.softwareforeducation.com/sms32v50/index.php Należy wybrać wersję 5. Nie ma znaczenia czy pobierzemy archiwum, czy jego wersję samorozpakowującą. Po rozpakowaniu uruchamiamy plik *sms32v50.exe*. W




Rysunek 1. Okno główne programu z otwartymi wszystkimi dodatkowymi oknami.




Rysunek 2. Widok pamięci symulatora.

katalogu z symulatorem znajduje się także dokumentacja w formacie HTML (uruchamiamy plik index.html).

Rysunek 1 przedstawia program z widocznymi oknami obrazującymi urządzenia wejściowe i wyjściowe. Okna te można włączać i wyłączać za pomocą przycisków z ikonami znajdującymi się w górnej części okna głównego.

SMS32 posiada 256 bajtów symulowanej pamięci RAM. Jej aktualna zawartość jest wyświetlana w odpowiednim oknie (zob. rys. 2). Pamięć można oglądać w trzech trybach: jako liczby szesnastkowe, kody ASCII i liczby szesnastkowe ze słowną reprezentacją instrukcji procesora. Zakres pamięci od adresu C0 do FF jest traktowany jako pamięć wyświetlacza. Zapisanie do tych komórek odpowiednich kodów ASCII dla znaków spowoduje wyświetlenie ich na wyświetlaczu VDU. Okno wyświetlacza powinno pokazać się automatycznie. Można je także wyświetlić ręcznie klikając ikonę monitora w oknie programu. Pamięć wyświetlacza traktowana jest także jak normalna pamięć programu. Można w niej zamieszczać instrukcje i dane, ale w takim przypadku na wyświetlaczu pokażą się „śmieci".

Kod w języku assemblera umieszcza się w edytorze pod zakładką „*Source code*". Można też kod wczytać. W katalogu z programem znajduje się klika przykładowych programów. Należy pamiętać, że zmieniony plik zostaje zapisany automatycznie po uruchomieniu programu lub jego asemblacji (przycisk „*Assemble*"). Program uruchamia się przyciskiem „*Run*" (wykonanie ciągłe) lub przyciskiem „*Step*" (wykonanie krokowe). Należy pamiętać, że liczby wpisywane w kodzie są traktowane jako szesnastkowe.

Tabela 1. Rejestry symulowanego mikroprocesora

| Rejestr | Funkcja |
|---|---|
| AL, BL, CL i DL | Rejestry ogólnego przeznaczenia |
| IP (ang. Instruction pointer) | Zawiera adres instrukcji, która będzie wykonywana lub jest aktualnie wykonywana. |
| SR (ang. Status Register) | Rejestr ten zawiera flagi, które informują o stanie procesora.<br>Flaga "Z" (zera) jest ustawiana na jeden, jeśli obliczenia dały wynik zerowy.<br>Flaga "S" (znaku) jest ustawiana na jeden, jeśli obliczenia dały wynik ujemny.<br>Flaga "O" (przepełnienia) jest ustawiana, jeśli wynik był zbyt duży, aby zmieścić się w rejestrze.<br>Flaga "I" (przerwania) jest ustawiana, jeśli przerwania są włączone. |
| SP (ang. Stack Pointer) | Wskaźnik stosu jest adresem kolejnego wolnego miejsca na stosie. |

Tabela 1 zawiera rejestry symulowanego układu. Są to rejestry 8-bitowe. Można je podejrzeć w oknie głównym programu. Wartości rejestrów są wyświetlane w systemie binarnym, szesnastkowym i dziesiętnym.

Tabela 2. Podstawowe instrukcje symulowanego mikroprocesora

| Instr. | Operand A | Operand B | Działanie |
|---|---|---|---|
| MOV | rejestr ogólnego przeznaczenia lub adres w pamięci. | konkretna wartość, rejestr ogólnego przeznaczenia lub adres w pamięci | kopiuje B do A. Operand docelowy i źródłowy nie mogą być jednocześnie adresami w pamięci ani rejestrami. |
| ADD | rejestr ogólnego przeznaczenia | konkretna wartość lub rejest | sumuje A i B; wynik przechowuje w A. |
| SUB | rejestr ogólnego przeznaczenia | konkretna wartość lub rejest | odejmuje B od A; wynik przechowuje w A. |
| MUL | rejestr ogólnego przeznaczenia | konkretna wartość lub rejesti | mnoży A i B; wynik przechowuje w A. |
| DIV | rejestr ogólnego przeznaczenia | konkretna wartość lub rejestr | dzieli A przez B; wynik przechowuje w A. |
| MOD | rejestr ogólnego przeznaczenia | konkretna wartość lub rejestr | dzieli A przez B; resztę z dzielenia przechowuje w A. |
| INC | rejestr ogólnego przeznaczenia | | zwiększa A o 1 |
| DEC | rejestr ogólnego przeznaczenia | | zmniejsza A o 1 |
| CMP | rejestr ogólnego przeznaczenia | konkretna wartość, rejestr lub adres pamięci | porównuje A i B ustawiając odpowiednio flagi |

Instrukcje w języku asemblera zapisujemy w postaci „*nazwa_instrukcji lista_operandów_oddzielonych_przecinkami*”. Operand to nic innego, jak po prostu argument danej instrukcji. W tabeli 2 zostały opisane podstawowe instrukcje, które będą wykorzystywane na dzisiejszych laboratoriach.

Listingi 1, 2 i 3 zawierają kody programów w języku asemblera wraz z komentarzami objaśniającymi działanie poszczególnych instrukcji. Zrozumienie działania tych instrukcji będzie niezbędne do rozwiązania zadań.

Listing 1. Program demonstrujący instrukcję kopiowania danych (MOV)

```
   CLO       ; zamknij wszystkie okna.

;  TRYB ADRESOWANIA NATYCHMIASTOWEGO
;  liczba hexadecymalna jest kopiowana do rejestru.
   MOV   AL,15 ; Kopiuj 15 HEX do rejestru AL
   MOV   BL,40 ; Kopiuj 40 HEX do rejestru BL
```

```
   MOV   CL,50 ; Kopiuj 50 HEX do rejestru CL
   MOV   DL,60 ; Kopiuj 60 HEX do rejestru DL

   INC   AL ; Zwieksz zawartosc rejestru AL.

;     TRYB ADRESOWANIA POSREDNIEGO
;  Kopiowanie do (z) komorki pamieci RAM.

   MOV   [A0],AL  ; Kopiuj zawartsc rejestru AL
         ;do komorki pamieci o adresie A0

   MOV   BL,[40]  ; Kopiuj zawartsc komorki pamieci
         ;o adresie 40 do rejestru BL


;    TRYB ADRESOWANIA POSREDNIEGO PRZEZ REJESTR

;  Kopiowanie do (z) komorki pamieci RAM.
;  adres komorki pamieci znajduje sie w rejestrze

   MOV   [CL],AL  ; Kopiuj zawartsc rejestru AL do komorki pamieci
         ; o adresie zapisanym w rejestrze CL
   MOV   BL,[CL]  ; Kopiuj zawartsc komorki o adresie w rejestrze CL
         ; do rejestru BL.


   END
```

Listing 2. Program demonstrujący instrukcje arytmetyczne

```
; Arytmetyka   ten program nie potrafi dzielic przez zero.

; Dodawanie
   MOV   AL,2  ; Przenies 2 Hex do rejestru AL
   MOV   BL,2  ; Przenies 2 Hex do rejestru BL
   ADD   AL,BL ; Do zawartosci AL dodaj zawartosc BL
         ; wynik zapisz w AL - tryb DIRECT
   ADD   AL,2  ; Do zawartosci AL dodaj liczbe 2 Hex
         ; wynik zapisz w AL - tryb IMMEDIATE
   INC   AL ; Inkrementacja inaczej dodawanie 1 do AL

; Dodawanie ktore powoduje przeniesienie
   MOV   AL,7F ; Przenies 7F Hex do rejestru AL
   MOV   BL,1  ; Przenies 1 Hex do rejestru BL
   ADD   AL,BL ; Do zawartosci AL dodaj zawartosc BL
         ; wynik zapisz w AL - tryb DIRECT
         ; Odpowiedz jest zla poniewaz wystapilo przeniesienie
```

```
; Odejmowanie
   MOV   AL,8  ; Przenies 8 Hex do rejestru AL
   MOV   BL,5  ; Przenies 5 Hex do rejestru BL
   SUB   AL,BL ; Od zawartosci AL odejmij zawartosc BL
         ; wynik zapisz w AL - tryb DIRECT
   SUB   AL,2  ; Od zawartosci AL odejmij liczbe 2 Hex
         ; wynik zapisz w AL - tryb IMMEDIATE
   DEC   AL ; Dekremenacja inaczej odejmowanie 1 od AL

; Odejmowanie z ustawianiem bitu znaku
   MOV   AL,5  ; Przenies 5 Hex do rejestru AL
   MOV   BL,8  ; Przenies 5 Hex do rejestru BL
   SUB   AL,BL ; Od zawartosci AL odejmij zawartosc BL
         ; wynik zapisz w AL - tryb DIRECT

; Mnozenie
   MOV   AL,2  ; Przenies 2 Hex do rejestru AL
   MOV   BL,2  ; Przenies 2 Hex do rejestru BL
   MUL   AL,BL ; Pomnoz AL * BL - wynik zapisz w AL - tryb DIRECT
   MUL   AL,2  ; Pomnoz AL * 2   - wynik zapisz w AL - tryb IMMEDIATE

; Mnozenie ktore powoduje przeniesienie
   MOV   AL,55 ; Przenies 55 Hex do rejestru AL
   MOV   BL,2  ; Przenies 2 Hex do rejestru BL
   MUL   AL,BL ; Pomnoz AL * BL - wynik zapisz w AL - tryb DIRECT
         ; Odpowiedz jest zla poniewaz wystapilo
         ; przeniesienie

; Dzielenie
   MOV   AL,10 ; Przenies 10 Hex do rejestru AL
   MOV   BL,2  ; Przenies 2 Hex do rejestru BL
   DIV   AL,BL ; Podziel AL przez BL - wynik zapisz w AL - tryb DIRECT
   DIV   AL,2  ; Podziel AL przez 2 - wynik zapisz w AL - tryb IMMEDIATE

; Reszta z dzielenia
   MOV   AL,B  ; Przenies B Hex do rejestru AL
   MOV   BL,2  ; Przenies 2 Hex do rejestru BL
   MOD   AL,BL ; Podziel AL przez BL - reszte zapisz w AL - tryb DIRECT
   MOD   AL,2  ; Podziel AL przez 2 - reszte zapisz w AL - tryb IMMEDIATE

; Dzielenie przez zero jest bledem
   MOV   AL,10 ; Przenies 10 Hex do rejestru AL
   MOV   BL,0  ; Przenies 0 Hex do rejestru BL
   DIV   AL,BL ; Podziel AL przez BL - wynik zapisz w AL - blad
```

```
   MOV   AL,10 ; Przenies 10 Hex do rejestru AL
   DIV   AL,0  ; Podziel AL przez 0 - wynik zapisz w AL - blad

   END      ; koniec programu
```

Listing 3. Program demonstrujący działanie pamięci wyświetlacza

```
   CLO      ; zamknij wszystkie okna.

;  Kopiowanie kodow znakow ASCII do pamieci wyswietlacza
   MOV   AL,41 ; Kopiuj kod znaku A do rejestru AL
   MOV   [C0],AL  ; Kopiuj kod znaku z rejestru do komorki C0
   MOV   AL,53 ; Kopiuj kod znaku S do rejestru AL
   MOV   [C1],AL  ; Kopiuj kod znaku z rejestru do komorki C1
   MOV   AL,4B ; Kopiuj kod znaku K do rejestru AL
   MOV   [C2],AL  ; Kopiuj kod znaku z rejestru do komorki C2

   END
```

### 1.2. Dodatkowe informacje

— https://pl.wikipedia.org/wiki/Mikroprocesor
— https://pl.wikipedia.org/wiki/Procesor
— https://pl.wikipedia.org/wiki/Asembler
— https://pl.wikipedia.org/wiki/Odpytywanie
— https://pl.wikipedia.org/wiki/Przerwanie
— https://pl.wikipedia.org/wiki/Rejestr_procesora

## 2. Zadania

Większość zadań wymaga wykonywania określonych instrukcji wiele razy. Ze względu na fakt, że konstrukcje pętli nie zostały jeszcze poznane, należy instrukcje wprowadzić wiele razy (najwygodniej kopiując).

### 2.1. Zadanie 1

Napisać program, który pod adresami [C0], [C1], [C2], [C3] i [C4] zapisze kody ASCII liter H,E,L,L,O. Po uruchomieniu programu, na wyświetlaczu VDU powinien wyświetlić się napis „HELLO”.

### 2.2. Zadanie 2

Napisać program obliczający sumę liczb z zakresu -4..3. Należy wykorzystać instrukcję inkrementacji (INC).

### 2.3. Zadanie 3

Napisać program obliczający sumę parzystych liczb z zakresu -11..5

### 2.4. Zadanie 4

Napisać program obliczający iloczyn liczb z zakresu 2..5

### 2.5. Zadanie 5

Napisać program obliczający $4^3$