Hardware: Procesor 8086 Konstrukcja i działanie

Procesor Intel 8086 jest mikroprocesorem 16-bitowym, który zadebiutował w 1978 roku. Był to jeden z pierwszych procesorów, które wprowadziły architekturę x86, która zyskała później popularność i stanowi podstawę wielu współczesnych procesorów. 8086 jest przykładem procesora CISC (Complex Instruction Set Computer), który posiada bogaty zestaw rozkazów, umożliwiający realizację skomplikowanych operacji w jednym cyklu.

Budowa i architektura procesora 8086

Procesor 8086 posiada szereg cech, które zdefiniowały jego architekturę:

* Rejestry : Procesor posiada 14 głównych rejestrów, w tym rejestry ogólnego przeznaczenia (AX, BX, CX, DX, SI, DI, BP, SP), rejestry segmentowe (CS, DS, SS, ES) oraz rejestr wskaźnika instrukcji (IP).

* Adresowanie pamięci : Procesor używa segmentowego adresowania pamięci, co oznacza, że cała pamięć 8086 jest podzielona na segmenty (do 1 MB pamięci). Adresy pamięci są tworzone przez 16-bitowy rejestr segmentu oraz 16-bitowy rejestr offsetu.

* Cykl pracy : 8086 działa na zasadzie cykli zegara, w których wykonuje operacje odczytu, zapisu i wykonywania instrukcji.

Instrukcje i tryby pracy

Procesor 8086 wykonuje instrukcje w różnych trybach:

* Tryb real-mode : Jest to tryb, w którym procesor wykonuje operacje bez wirtualizacji pamięci i bez ochrony pamięci.

* Tryb ochrony: Tryb ten jest bardziej zaawansowany i pozwala na zarządzanie pamięcią w sposób bezpieczniejszy (chociaż procesor 8086 nie obsługiwał go bezpośrednio, ale w późniejszych wersjach, jak 80286, ten tryb został wprowadzony).

Przykłady programów w assemblerze procesora 8086

Poniżej przedstawiam kilka przykładów programów w języku assemblera dla procesora 8086. Przykłady te ilustrują podstawowe operacje wykonywane przez procesor, jak manipulacja rejestrami, wejście/wyjście na porty oraz podstawowe skoki warunkowe.

Program 1

```
JMP main        ;
```

```
    DB 90        ;
    DB D8        ;
    DB 24        ;
    DB 48        ;

main:
    MOV DL,02    ; ustawienie DL na pierwszym adresie danych w RAM
            ; Turn off all the traffic lights.
    MOV AL,0    ; Copy 00000000 into the AL register.
    OUT 01        ; Send AL to Port One (The traffic lights).

start:
    MOV AL,[DL]    ; pobranie wartości spod adresu DL
    OUT 01        ;
    INC DL        ;
    CMP DL,5    ;
    JZ end        ;
    JMP start    ; Jump back to the start.
end:
    END        ; Program ends.
```

Opis programu

* Program wczytuje dane z pamięci i wysyła je na port (symulując sterowanie sygnalizacją świetlną).

* Wartości są przechowywane w pamięci, a procesor przesyła je na odpowiedni port za pomocą instrukcji `OUT`.

* Program zatrzymuje się, gdy liczba wysłanych danych osiągnie 5.

Program 2

```
Start:
    MOV    AL,B6    ; 1111 1010
    OUT    02    ; Send the data in AL to Port 02

    MOV    AL,0B    ; 0000 0000
    OUT    02    ; Send the data in AL to Port 02

    JMP    Start

    END
```

Opis programu

* Program ustawia wartość `B6` w rejestrze AL, a następnie wysyła ją na port 02.

* Następnie ustawia wartość `0B` w rejestrze AL i ponownie wysyła ją na port 02.

* Program powtarza te operacje w nieskończoność.

Program 3

```
start:

    IN    03    ; Input from Port 03
    AND    AL,1    ; Mask off left seven bits
    JZ    Cold    ; If the result is zero, turn the heater on
    JMP    Hot    ;

Cold:
    MOV    AL,80    ; Code to turn the heater on
    OUT    03    ; Send code to the heater
    JMP start    ;
Hot:
    MOV    AL,0    ; Code to turn the heater off
    OUT    03    ; Send code to the heater
    JMP start    ;

    JMP start    ;

END
```

Opis programu

* Program odczytuje dane z portu 03 i sprawdza, czy najmłodszy bit jest ustawiony na 0 (co może oznaczać, że temperatura jest zbyt niska).

* W zależności od wyniku, program włącza lub wyłącza grzałkę, wysyłając odpowiedni kod na port 03.

Program 4

```
JMP Start

    DB    1        ;
    DB     "I AM KACPER"     ;
```

```
    DB     0        ;


Start:

    MOV     BL,02         ; 02 adres początku danych
    MOV     CL,C0         ; C0 adres początku danych wyświetlacza

Rep1:

    MOV     AL,[BL]        ; z komórki w pamięci o adresie BL pobieram
wartość do rejestru AL
    MOV     DL,[BL]        ; z komórki w pamięci o adresie DL pobieram
wartość do rejestru AL
    PUSH     AL        ;
    MOV     [CL],DL        ; z rejestru DL zapisuje wartość w pamięci o
adresie CL
    CMP     AL,0         ; czy ostatni element ? 5 to ostatni adres komórki
w której są dane
    JZ     continue     ; jesli tak, to skocz do start, zaczynamy od
poczatku
    INC     BL         ; jesli nie, to wez kolejny element
    INC     CL        ; zwiększam adres danych wyświetlacza
    JMP     Rep1         ; skok na poczatek

continue:

    MOV     CL,D0         ; C0 adres początku danych wyświetlacza

Rep2:

    POP     AL        ;
    MOV     [CL],AL        ; z rejestru DL zapisuje wartość w pamięci o
adresie CL
    CMP     AL,1         ; czy ostatni element ? 5 to ostatni adres komórki
w której są dane
    JZ     end         ; jesli tak, to skocz do start, zaczynamy od poczatku
    INC     BL         ; jesli nie, to wez kolejny element
    INC     CL        ; zwiększam adres danych wyświetlacza
    JMP     Rep2         ; skok na poczatek

end:
    end
```

Opis programu

* Program wyświetla napis „I AM KACPER” na ekranie.

* Przesuwa dane z jednej komórki pamięci do drugiej, wykorzystując rejestry i instrukcje manipulacji

pamięcią.

Symulator SMZ32V50

Do wykonywania powyższych programów użyty został symulator SMZ32V50. Jest to narzędzie, które umożliwia emulację działania procesora 8086, pozwalając na testowanie i debugowanie programów w assemblerze. Symulator ten może emulować działanie portów wejścia/wyjścia oraz pamięci, co czyni go idealnym narzędziem do nauki i testowania kodu dla tego typu procesorów.

smz32v50.exe

Podsumowanie

Procesor 8086 był pionierem w wielu aspektach architektury komputerowej. Dzięki jego prostocie i elastyczności, programowanie w assemblerze tego procesora wciąż stanowi cenne źródło nauki dla tych, którzy chcą zrozumieć podstawy działania komputerów. Przykłady powyższych programów pokazują, jak wykorzystywać instrukcje procesora do realizacji różnych operacji, takich jak manipulacja rejestrami, interakcje z portami wejścia/wyjścia oraz proste struktury sterujące.
