Hardware: 8086 processor Design and operation

The Intel 8086 processor is a 16-bit microprocessor that debuted in 1978. It was one of the first processors to introduce the x86 architecture, which later became popular and forms the basis of many modern processors. The 8086 is an example of a CISC (Complex Instruction Set Computer) processor, which has a rich instruction set that allows complex operations to be performed in a single cycle.

Structure and architecture of the 8086 processor

The 8086 processor has a number of features that have defined its architecture:

* Registers : The processor has 14 main registers, including general purpose registers (AX, BX, CX, DX, SI, DI, BP, SP), segment registers (CS, DS, SS, ES) and an instruction pointer (IP) register.

* Memory addressing : The processor uses segmented memory addressing, which means that all 8086 memory is divided into segments (up to 1 MB of memory). Memory addresses are created by a 16-bit segment register and a 16-bit offset register.

* Duty cycle : The 8086 operates on the principle of clock cycles, in which it performs read, write and instruction execution operations.

Instructions and modes of operation

The 8086 processor executes instructions in various modes:

* Real-mode : This is the mode in which the processor performs operations without memory virtualisation and without memory protection.

* Protection mode : This mode is more advanced and allows memory to be managed in a more secure way (although the 8086 processor did not support it directly, but in later versions, such as the 80286, this mode was introduced).

Examples of programs in the 8086 processor assembler

Below I present some examples of assembler language programs for the 8086 processor. These examples illustrate basic operations performed by the processor, such as register manipulation, port input/output and basic conditional jumps.

Programme 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.
```

Programme description

* The programme loads data from memory and sends it to the port (simulating traffic light control).

* Values are stored in memory and the processor sends them to the corresponding port using an `OUT` instruction.

* The program stops when the number of data sent reaches 5.

Programme 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
```

Description of the programme

* The program sets the value `B6` in the AL register and then sends it to port 02.

* It then sets the value `0B` in the AL register and sends it to port 02 again.

* The program repeats these operations indefinitely.

Programme 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
```

Programme description

* The program reads data from port 03 and checks that the youngest bit is set to 0 (which may mean that the temperature is too low).

* Depending on the result, the program turns the heater on or off by sending the appropriate code to 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
```

Programme description

* The programme displays the words „I AM KACPER” on the screen.

* Moves data from one memory cell to another using registers and memory manipulation instructions.

SMZ32V50 simulator

The SMZ32V50 simulator was used to execute the above programs. It is a tool that enables the emulation of the operation of the 8086 processor, allowing the testing and debugging of programs in assembler. This simulator can emulate the operation of input/output ports and memory, making it an ideal tool for learning and testing code for this type of processor.

smz32v50.exe

Summary

The 8086 processor was a pioneer in many aspects of computer architecture. Thanks to its simplicity and flexibility, programming in the assembler of this processor is still a valuable learning resource for those who want to understand the basics of computers. The examples of the above programs show how to use the processor's instructions to perform a variety of operations such as register manipulation, interaction with input/output ports and simple control structures.
