Materials

Ćw1 Bardzo prosty model komputera (BPMK), jego język maszynowy i assembler.

Omawiane zagadnienia j.w.

Very Simple Machine, its Machine Language and Assembler

Very Simple Machine

- External memory of 10000 memory cell.
- Each memory cell can store one five-digit number.
- Decimal numbers.
- 1 accumulator (internal memory).

Accumulator is a dedicated memory cell located in CPU. Such dedicated memory cells are also called **register(s)**. Memory (RAM -- *random access memory*) consist of 10000 cells with numbers (**addresses**) from 0 to 9999. A sign-value representation is used to store negative/positive numbers -- when most significant digit is set to 0, the number is positive and negative otherwise (i.e. when different than 0). All arithmetic instructions works on signed numbers.

Machine Language and Assembler

Machine Language vs. Assembly Language: Key Differences

Machine language and assembly language are both low-level programming languages. The main difference is that assembly language is a symbolic representation of machine language, consisting of binary code executed directly by the computer's hardware.

Machine code, object code, or machine language is a collection of bits (or binary digits) to be read and interpreted by a computer. It is the only language understood by a computer.

On the other hand, assembler language, symbolic machine code, or assembly language is

any low-level programming language in computer programming with a high level of correspondence between the language instructions and the machine code instructions of the architecture.

You will use decimal numbers and 5 digit instruction of the following format

```
operation code (5th digit; the most significant digit)
|
xxxxx
 |  |
 +--+
    |
    opernad(s) (digits from 4th to 1st;
                1st is the least significant digit)
```

The list of instruction is as follow (first **operation code**, opcode, is given next **mnemonic**):

```
0 HLT stop the cpu
1 CPA copy value from memory to accumulator, M -> A
2 STO copy value from accumulator to memory, A -> M
3 ADD add value from specified memory cell to accumulator;
      result is stored in accumulator, M + A -> A
4 SUB subtract from accumulator value from specified memory cell;
      result is stored in accumulator A - M -> A
5 MUL multiply value from accumulator by value from specified memory cel
      result is stored in accumulator M * A -> A
6 BRA unconditional branch to instruction located at specified address
7 BRN conditional branch to instruction located at specified address
      if value stored in accumulator is negative
8 BRZ conditional branch to instruction located at specified address
      if value stored in accumulator is equal to zero
```

Instruction number 9 is reserved for future extensions.

Excercises

Excercise 1

Write a program to calculate sum of numbers 20, 30 and 40 located at address 6, 7 and 8 respectively; result store at address 9.

```
Address Value
0006    20
0007    30
0008    40
0009    ?         ; Result
```

If `?` is used in place of real data it means that real value is not important at the moment of starting the code (it can be anything random), because it would be overwritten in the future before any atempt of reading.

Solution 1.1

```
        Machine
        language    Assembler

Address Value       Instruction    Comment
0010    10006       CPA 6          ; A=20
0011    30007       ADD 7          ; A=20+30
0012    30008       ADD 8          ; A=20+30+40
0013    20009       STO 9
0014    00000       HLT
```

Excercise 2

Write a program to calculate for a given x a value of polynomial P:

$$P(x) = ax + b$$

Data arrangement in memory

```
Address Value
0004    ?         ; Result
0005    x         ; For example x = 2
0006    a         ; For example a = 3
0007    b         ; For example b = 4
```

Solution 2.1

```
Address Value       Instruction    Comment
0010    10006       CPA 6          ; A=3
0011    50005       MUL 5          ; A=3*2
0012    30007       ADD 7          ; A=3*2+4
0013    20004       STO 4          ; Copy A to address 4
0014    00000       HLT            ; Stop
```

Excercise 3

Write a program to calculate for a given x a value of polynomial P:

$$P(x) = ax^3 + bx^2 + cx + d$$

Data arrangement in memory

```
Address Value
0004    ?          ; Result
0005    x          ; For example x = 2
0006    a          ; For example a = 3
0007    b          ; For example b = 4
0008    c          ; For example c = 5
0009    d          ; For example d = 6
```

Solution 3.1

```
Address Value       Instruction  Comment
0010    10005       CPA 5        ; Copy x to accumulator (A=x)
0011    50005       MUL 5        ; Multiply A by x, A=x^2
0012    50005       MUL 5        ; Multiply A by x, A=x^3
0013    50006       MUL 6        ; Multiply A by a, A=x^3*a
0014    20004       STO 4        ; Copy A to address 4 (result)
0015    10005       CPA 5        ; Copy x to accumulator (A=x)
0016    50005       MUL 5        ; Multiply A by x, A=x^2
0017    50007       MUL 7        ; Multiply A by b, A=x^2*b
0018    30004       ADD 4        ; Add to A value from result
0019    20004       STO 4        ; Copy A to address 4 (result)
0020    10005       CPA 5        ; Copy x to accumulator (A=x)
0021    50008       MUL 8        ; Multiply A by c, A=x*c
0022    30004       ADD 4        ; Add to A value from address 4 (result
0023    20004       STO 4        ; Copy A to address 4 (result)
0024    10009       CPA 9        ; Copy d to accumulator (A=x)
0025    30004       ADD 4        ; Add to A value from address 4 (result
0026    20004       STO 4        ; Copy A to address 4 (result)
0027    00000       HLT          ; Stop
```

Solution 3.2

```
Address Value      Instruction  Comment
0010    10005       CPA 5       ; Copy x to accumulator (A=x)
0011    50005       MUL 5       ; Multiply A by x, A=x^2
0012    50005       MUL 5       ; Multiply A by x, A=x^3
0013    50006       MUL 6       ; Multiply A by a, A=x^3*a
0014    20100       STO 100     ; Copy A to address 100
0015    10005       CPA 5       ; Copy x to accumulator (A=x)
0016    50005       MUL 5       ; Multiply A by x, A=x^2
0017    50007       MUL 7       ; Multiply A by b, A=x^2*b
0018    20101       STO 101     ; Copy A to address 101
0019    10005       CPA 5       ; Copy x to accumulator (A=x)
0020    50008       MUL 8       ; Multiply A by c, A=x*c
0021    20112       STO 102     ; Copy A to address 102
0022    10009       CPA 9       ; Copy d to accumulator (A=d)
0023    30100       ADD 100     ; Add x^3*a to accumulator (A=x^3*a+d)
0024    30111       ADD 101     ; Add x^2*b to accumulator (A=x^3*a+x^2*
0025    30112       ADD 102     ; Add x*c to accumulator (A=x^3*a+x^2*b+
0026    20004       STO 4       ; Copy A to address 4 result
0027    00000       HLT         ; Stop
```

Solution 3.3

In this solution I use a **Horner's rule**, in which a polynomial is written in nested form:

$$a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n =$$

$$a_0 + x\left(a_1 + x\left(a_2 + x\left(a_3 + \cdots + x(a_{n-1} + x\,a_n)\cdots\right)\right)\right)$$

```
Address Value       Instruction    Comment
0010    10006       CPA 6          ; A=a
0011    50005       MUL 5          ; A=ax
0012    30007       ADD 7          ; A=ax + b
0013    50005       MUL 5          ; A=(ax + b)x
0014    30008       ADD 8          ; A=(ax+b)x+c
0015    50005       MUL 5          ; A=((ax+b)x+c)x
0016    30009       ADD 9          ; A=((ax+b)x+c)x+d
0017    20004       STO 4          ; Copy A to address 4 result
0018    00000       HLT            ; Stop
```

Excercise 4

Calculate a^b , where a -- integer number, b -- integer nonnegative number.

Data arrangement in memory

```
Address Value
0001    a
0002    b
```

Solution 4.1

```
Address Value       Instruction  Comment
0001    xxxxx       a
0002    xxxxx       b
0003    00001       1            ; Iterator
0004    xxxxx       ?            ; Result
0005    10003       CPA 3        ; Copy 1 to A
0006    20004       STO 4        ; Copy A to result
0007    10002       CPA 2        ; Copy b to A
0008    80015       BRZ 15       ; Jump to 15 if A=b is zero
0009    40003       SUB 3        ; Subtract 1 from A=b
0010    20002       STO 2        ; Copy A to b (A=b-1)
0011    10004       CPA 4        ; Copy result to A
0012    50001       MUL 1        ; Multiply A=result by a
0013    20004       STO 4        ; Copy A to result
0014    10002       CPA 2        ; Copy b to A
0014    80007       BRZ 7        ; Jump to 7 if A=b < 0
0015    00000       HLT          ; Stop
```

Solution 4.2

```
Address Value       Instruction  Comment
0001    xxxxx       a
0002    xxxxx       b
0003    00001       1             ; Iterator
0004    00001       ?             ; Result
0005    10002       CPA 2         ; Copy b to A
0006    80013       BRN 13        ; Jump if b<0
0007    40003       SUB 3         ; Subtract iterator from b
0008    20002       STO 2         ; Save iterator
0009    10004       CPA 4         ; Copy result to A
0010    50001       MUL 1         ; Multiply A by a
0011    20004       STO 4         ; Save as result
0012    60005       BRA 5         ; End loop - jump to the begining of t
0013    00000       HLT           ; Stop
```

Excercise 5

Calculate $\frac{a}{b}$, where both a and b is an integer positive number.

```
Address Value
0001    a
0002    b
0003    ?         ; Result (integer part)
0004    ?         ; Result (fractional part)
```

An integer part of division is stored at address 0003, fractional part at address 0004.

Solution 5.1

```
Address Value       Instruction  Comment
0001    xxxxx       a
0002    xxxxx       b
0003    0           result (integer part)
0004    xxxxx       result (fractional part)

0005    00001                    ; Constant value for counter

                                 ; Main part
0006    10001       CPA 1        ; Copy a to A
0007    40002       SUB 2        ; Subtract b
0008    10002       BRN 16       ; If A is negative than go to the end
0009    20001       STO 1        ; Copy a-b to a
                                 ; Increment integer part
0010    10003       CPA 3        ; Copy integer part to A
0011    30005       ADD 5        ; Add constant 1 to A
0012    20003       STO 3        ; Save incremented integer part
                                 ; End increment
0013    60010       BRA 6        ; Go to the begining of the loop

0014    10001       CPA 1        ; Copy a to A
0015    20004       STO 4        ; Copy A=a to fractional part
0016    00000       HLT          ; Stop
```