COMP303 - Computer Architecture

Homework 2: (return by 31.10.2018).

PS: HWs submitted after the deadline will not be accepted

Email your HW to: nizamettinaydin@gmail.com

You should place CAr-hw2 in the "Subject" field.

You must name your file by using your matriculation number

(ie, hw2-53745.vvm or hw2-53745.txt)

The following code is a portion of a C program. A is an array with length of N.

```
for (int n = 0; n < N; n++ )
{
    B(n) = abs( A(N-1-n));
}
```

If A is given as follows

A = {-1,3,-5,-2,4,-6,5,-3,1,-9}

using the VVM instruction set, write an assembler program that implements the C code given above.

(Assume that array A starts at address 60, array B starts at address 70)

VVM instruction set

(*nn* indicates an address location in the RAM), for example;
machine code **510** corresponds to Assembly code **LDA 10** (or **lda 10**).

Load Accumulator (5*nn*) [LDA *nn*] The content of RAM address *nn* is copied to the Accumulator Register, replacing the current content of the register. The content of RAM address *nn* remains unchanged. The Program Counter Register is incremented by one.

Store Accumulator (3*nn*) [STO *nn*] or **[STA *nn*]** The content of the Accumulator Register is copied to RAM address *nn*, replacing the current content of the address. The content of the Accumulator Register remains unchanged. The Program Counter Register is incremented by one.

Add (1*nn*) [ADD *nn*] The content of RAM address *nn* is added to the content of the Accumulator Register, replacing the current content of the register. The content of RAM address *nn* remains unchanged. The Program Counter Register is incremented by one.

Subtract (2*nn*) [SUB *nn*] The content of RAM address *nn* is subtracted from the content of the Accumulator Register, replacing the current content of the register. The content of RAM address *nn* remains unchanged. The Program Counter Register is incremented by one.

Input (901) [IN] or **[INP]** A value input by the user is stored in the Accumulator Register, replacing the current content of the register. Note that the two-digit operand does not represent an address in this instruction, but rather specifies the particulars of the I/O operation (see Output). The operand value can be omitted in the Assembly Language format. The Program Counter Register is incremented by one with this instruction.

Output (902) [OUT] or **[PRN]** The content of the Accumulator Register is output to the user. The current content of the register remains unchanged. Note that the two-digit operand does not represent an address in this

instruction, but rather specifies the particulars of the I/O operation (see Input). The operand value can be omitted in the Assembly Language format. The Program Counter Register is incremented by one with this instruction.
Branch if Zero (7*nn*) [BRZ *nn*] This is a conditional branch instruction. If the value in the Accumulator Register is zero, then the current value of the Program Counter Register is replaced by the operand value *nn* (the result is that the next instruction to be executed will be taken from address *nn* rather than from the next sequential address). Otherwise (Accumulator >< 0), the Program Counter Register is incremented by one (thus the next instruction to be executed will be taken from the next sequential address).
Branch if Positive or Zero (8*nn*) [BRP *nn*] This is a conditional branch instruction. If the value in the Accumulator Register is nonnegative (i.e., >= 0), then the current value of the Program Counter Register is replaced by the operand value *nn* (the result is that the next instruction to be executed will be taken from address *nn* rather than from the next sequential address). Otherwise (Accumulator < 0), the Program Counter Register is incremented by one (thus the next instruction to be executed will be taken from the next sequential address).
Branch (6*nn*) [BR *nn*] or**[BRU *nn*]** or **[JMP *nn*]** This is an unconditional branch instruction. The current value of the Program Counter Register is replaced by the operand value *nn*. The result is that the next instruction to be executed will be taken from address *nn* rather than from the next sequential address. The value of the Program Counter Register is not incremented with this instruction.
No Operation (4*nn*) [NOP] or **[NUL]** This instruction does nothing other than increment the Program Counter Register by one. The operand value *nn* is ignored in this instruction and can be omitted in the Assembly Language format. (This instruction is unique to the VVM and is not part of the Little Man Model.)
Halt (0*nn*) [HLT] or **[COB]** Program execution is terminated. The operand value *nn* is ignored in this instruction and can be omitted in the Assembly Language format.

Embedding Data in Programs
Data values used by a program can be loaded into memory along with the program. In Machine or Assembly Language form simply use the format "*snnn*" where *s* is an optional sign, and *nnn* is the three-digit data value. In Assembly Language, you can specify "DAT *snnn*" for clarity.
