## Exercise 01

Examine the following program:

class Program

{

static void Main(string[] args) {

Program prog = new Program();

prog.Run();

}

private void Run() {

//Loop(1);

Recursion(1);

}

private void Loop(int v) {

while(v > 0) {

Console.WriteLine(v);

v++;

}

}

private void Recursion(int v) {

Console.WriteLine(v);

Recursion(++v);

}

}

Obviously there are some problems with the two methods, Loop(int v) and Recursion(int v) – but that is intended.

* Figure out what the result will be when running either of the methods.

Now, copy-paste the program into a C# solution and run first one of the methods then the other (out-comment them in turn). Have each method count to at least 60.000 (if possible…).

* Was the result as you predicted?
* How do you explain what happened?

## CPU simulator

Imagine a very simple computer system that consists only of a CPU and RAM:

* There are only 40 RAM addresses
* The CPU only has two registers: the *accumulator* and the *program counter*
* The CPU only has 10 different instructions
* Only integers can be used – no decimals available
* The accumulator can only hold positive integers

An instruction is using 6 digits: the first two identifies the opcode, the last four holds the operands (if not used they are zero). The instructions are explained in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Instruction mnemonic | Explanation | Example |
| 01 | LOAD | Loads a value from RAM into the accumulator | 010034 – if value in address 34 is 7, the value in accumulator now also is 7 |
| 02 | ADD | Adds a value from RAM to the value in the accumulator; stores the result in the accumulator | 020023 – if value in address 23 is 11 and value in accumulator was 3, the value in accumulator now is 14 |
| 03 | SUBTRACT | Subtracts a value from RAM from the value in the accumulator; stores the result in the accumulator | 030036 – if value in address 36 is 22 and accumulator was 41, the value in accumulator now is 19 |
| 04 | DIVIDE | Divides (using integer division) the value in the accumulator with a value from RAM; stores the result in the accumulator | 040015 – if value in address 15 is 3 and value in accumulator was 13, the value in accumulator now is 4 |
| 05 | MULTIPLY | Multiplies the value in the accumulator by a value from RAM; stores the result in the accumulator | 050038 – if value in address 38 is 4 and value in accumulator was 14, value in accumulator now is 56 |
| 06 | STORE | Stores the value in the accumulator into an address in RAM | 060029 – if accumulator is 26, the value in address 29 now is also 26 |
| 07 | JUMP | The next instruction executed is from an address in RAM given by operand | 070029 – next instruction executed will be the one from address 29 (program counter “jumps” to address 29) |
| 08 | TEST ZERO | Test if accumulator holds zero; if result is true the next instruction is executed – if result is false the next instruction is skipped | 080000 – if accumulator holds 0 (zero) next instruction is executed; if accumulator holds value different from 0 (zero) next instruction is skipped |
| 09 | TEST GREATER THAN | Test if accumulator hold as value greater than (or equal to) a value from RAM; if result is true the next instruction is executed – if result is false the next instruction is skipped | 090032 – if value in accumulator is 19 and value in address 32 is 12 next instruction is executed. If value in accumulator is 19 and value in address 32 is 20 next instruction is skipped. |
| 99 | STOP | Ends the program | 990000– programs terminates. |

The file “*cpusim1\_2.html*” implements a system as described above.

Let us make a program that adds two numbers and stores the result in a RAM address. As there is no input or output available, we must store all needed values in the program – the values will be *hard-coded*!

Therefore, we do some planning:

* we will store the numbers to be added at RAM addresses 20 and 21
* we want the result to be stored in RAM address 22

Here is the program:

|  |  |
| --- | --- |
| Instruction | Explanation |
| 010020 | Load value (first number) from address 20 into accumulator |
| 020021 | Add value from address 21 (second number), store result in accumulator |
| 060022 | Store value from accumulator into address 22 |
| 990000 | End the program |

Our program takes four instructions. We load this program into RAM starting at address 00 and put the numbers 100 and 123 in addresses 20 and 21:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 00 | 010020 | 10 | 000000 | 20 | 000100 | 30 | 000000 |
| 01 | 020021 | 11 | 000000 | 21 | 000123 | 31 | 000000 |
| 02 | 060022 | 12 | 000000 | 22 | 000000 | 32 | 000000 |
| 03 | 990000 | 13 | 000000 | 23 | 000000 | 33 | 000000 |
| 04 | 000000 | 14 | 000000 | 24 | 000000 | 34 | 000000 |
| 05 | 000000 | 15 | 000000 | 25 | 000000 | 35 | 000000 |
| 06 | 000000 | 16 | 000000 | 26 | 000000 | 36 | 000000 |
| 07 | 000000 | 17 | 000000 | 27 | 000000 | 37 | 000000 |
| 08 | 000000 | 18 | 000000 | 28 | 000000 | 38 | 000000 |
| 09 | 000000 | 19 | 000000 | 29 | 000000 | 39 | 000000 |

At program-start the CPUs registers holds these values:

|  |  |
| --- | --- |
| Program Counter | Accumulator |
| 00 | 0 |

After ***first*** instruction (010020 at address 00), the CPUs registers holds these values:

|  |  |
| --- | --- |
| Program Counter | Accumulator |
| 01 | 100 |

After ***second*** instruction (020021 at address 01), the CPUs registers holds these values:

|  |  |
| --- | --- |
| Program Counter | Accumulator |
| 02 | 223 |

After ***third*** instruction (060022 at address 02), the CPUs registers holds these values:

|  |  |
| --- | --- |
| Program Counter | Accumulator |
| 03 | 223 |

In addition, RAM address 22 now has changed:

|  |  |
| --- | --- |
| 22 | 000223 |

After ***fourth*** instruction (990000 at address 03), the program ends.

## Exercise 02

Create a program that subtracts two numbers from each other and stores the result in a RAM address.

## Exercise 03

Create a program that subtracts two numbers from each other. Then it tests whether the result is greater than a third number – if so the result should be stored in address 30, else it should be stored in address 31

## Exercise 04

Create a program that calculates the double of a given number and stores the result in a RAM address

## Exercise 05

Use think-pair-share (3-2-2 minutes)

Read the following code and figure (*think*) out what happens (do ***not*** use the CPU simulator; only pen and paper).

Discuss your result with your shoulder partner.

Discuss the results in the group.

## Exercise 06 – Warning: hard!

Create a program that:

* reads a value from memory
* loops a given number of times
* in each iteration subtracts 4
* after the loop tests whether the result is greater than 10

## Exercise 07 – Warning: very hard!

Create a program has a “method”. The method takes one parameter, doubles the value and returns the result.

Run the program with 3 calls to the “method” using 3 different values. The sum of the 3 return values should be stored in RAM address 39.

Take care and plan well. First, you must consider carefully what is needed for a method to be called:

* how does the method access the parameter(s)?
* how does the method return a value to the caller?
* how does the method know where to continue after the method has ended (the method can be called from anywhere)?
* What should the caller do and what should the method do for all this to work?