VerySimpleCPU
Instruction Set Simulator (ISS)

- VerySimpleCPU.exe is an ISS (i.e., simulation model) written in C.
    - Takes a program file, coded in VerySimpleCPU assembly language, as input,
    - Reads VerySimpleCPU instructions,
    - Executes the instructions, and
    - Writes results on to the command window as well as text files.
- Provides a debugging environment for VerySimpleCPU programmers.

How to run:

1. Download everything through the link at:
   https://www.dropbox.com/sh/c9w08c07f2vuef5/AAAeubBdLv_x5OKzEIGjscPMa?dl=0

2. Open a cmd window
3. Type:
   ```
   VerySimpleCPU programFileName.asm
   ```
   then type r
4. You may also type:
   ```
   VerySimpleCPU programFileName.asm > log
   ```
   then type r

- Input File:
  - VerySimpleCPU assembly program file
- Output Files (in relation to simulation). :
  - memin.txt: Contains the memory contents before execution of the program.
  - memoutd.txt: After execution of the program, contains the memory contents in decimal.
  - memouth.txt: After execution of the program, contains the memory contents in hexadecimal.
- In addition, the simulator prints (on the screen) the memory contents before and after execution of each instruction.

Example: Summation

- A code written in a high-level programming language:

*103 = *100 + *101 + *102

↓ ↓ ↓

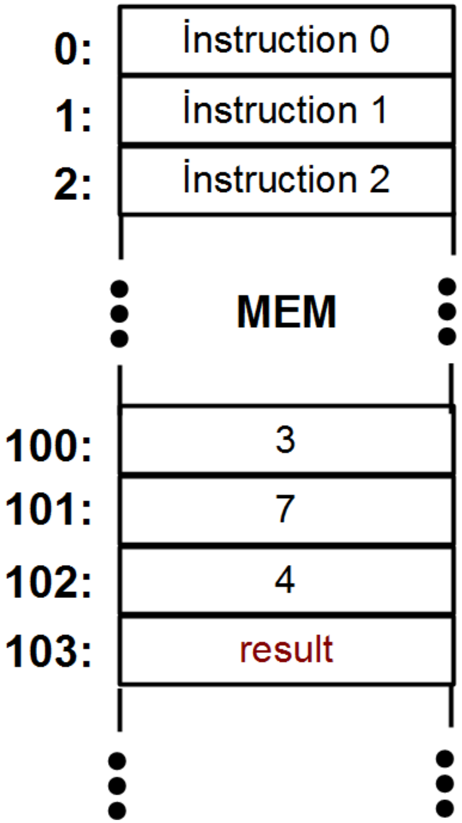
3 7 4

- Can be also written as

*103 = *100

*103 = *103 + *101

*103 = *103 + *102

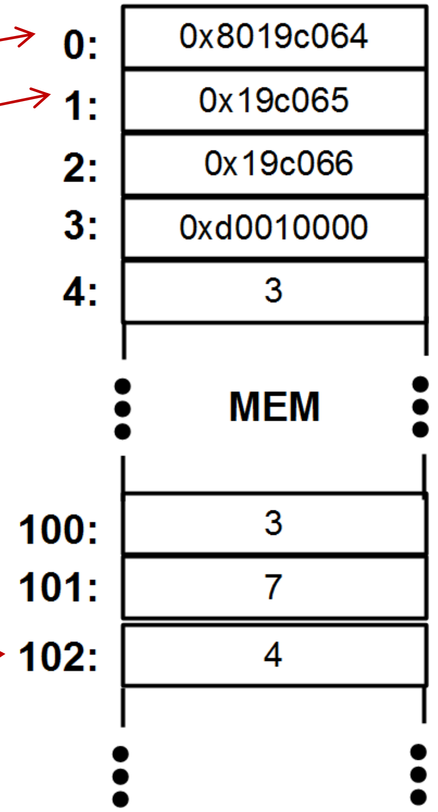


0:	İnstruction 0
1:	İnstruction 1
2:	İnstruction 2
⋮	MEM
100:	3
101:	7
102:	4
103:	result
⋮	

Example: (cont'd)

- The program written in VerySimpleCPU assembly language (sample.asm):

```
0: CP 103 100
1: ADD 103 101
2: ADD 103 102
3: BZJi 4 0
4: 3

100: 3
101: 7
102: 4
```



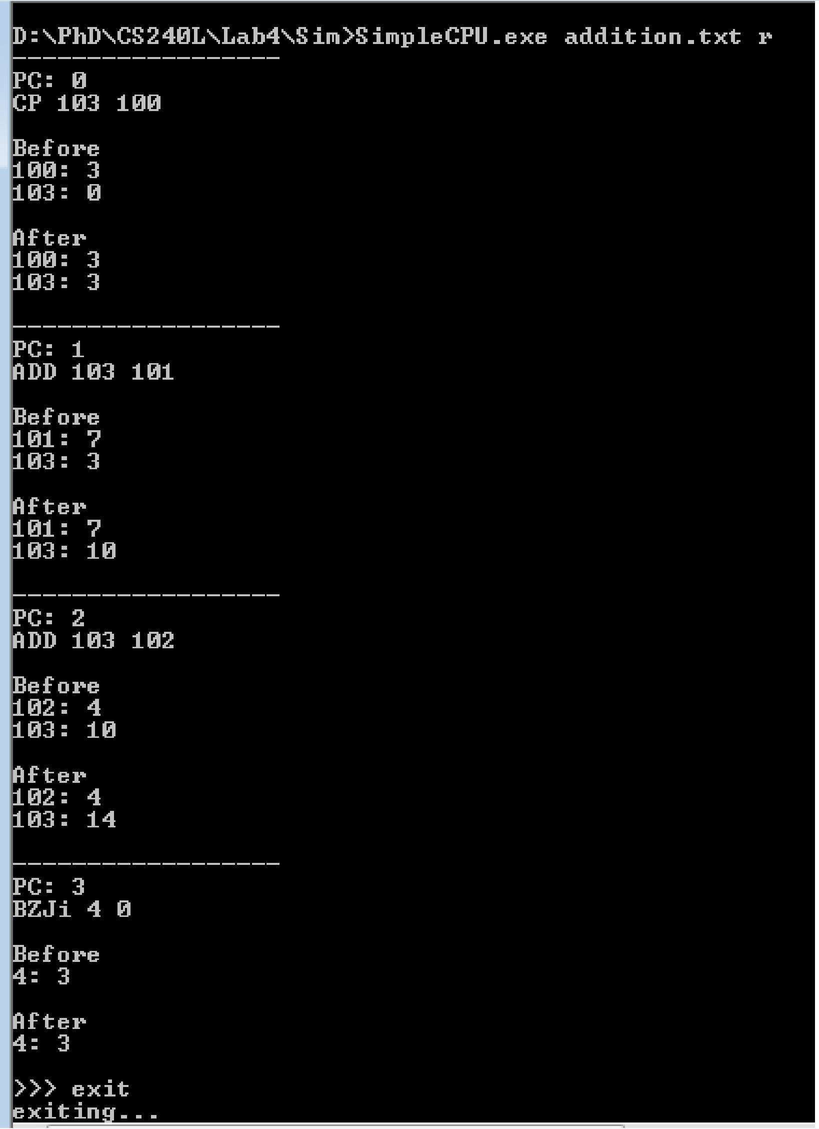
	MEM
0:	0x8019c064
1:	0x19c065
2:	0x19c066
3:	0xd0010000
4:	3
⋮	⋮
100:	3
101:	7
102:	4
⋮	⋮

Example: (cont'd)

- Running the ISS at command line:

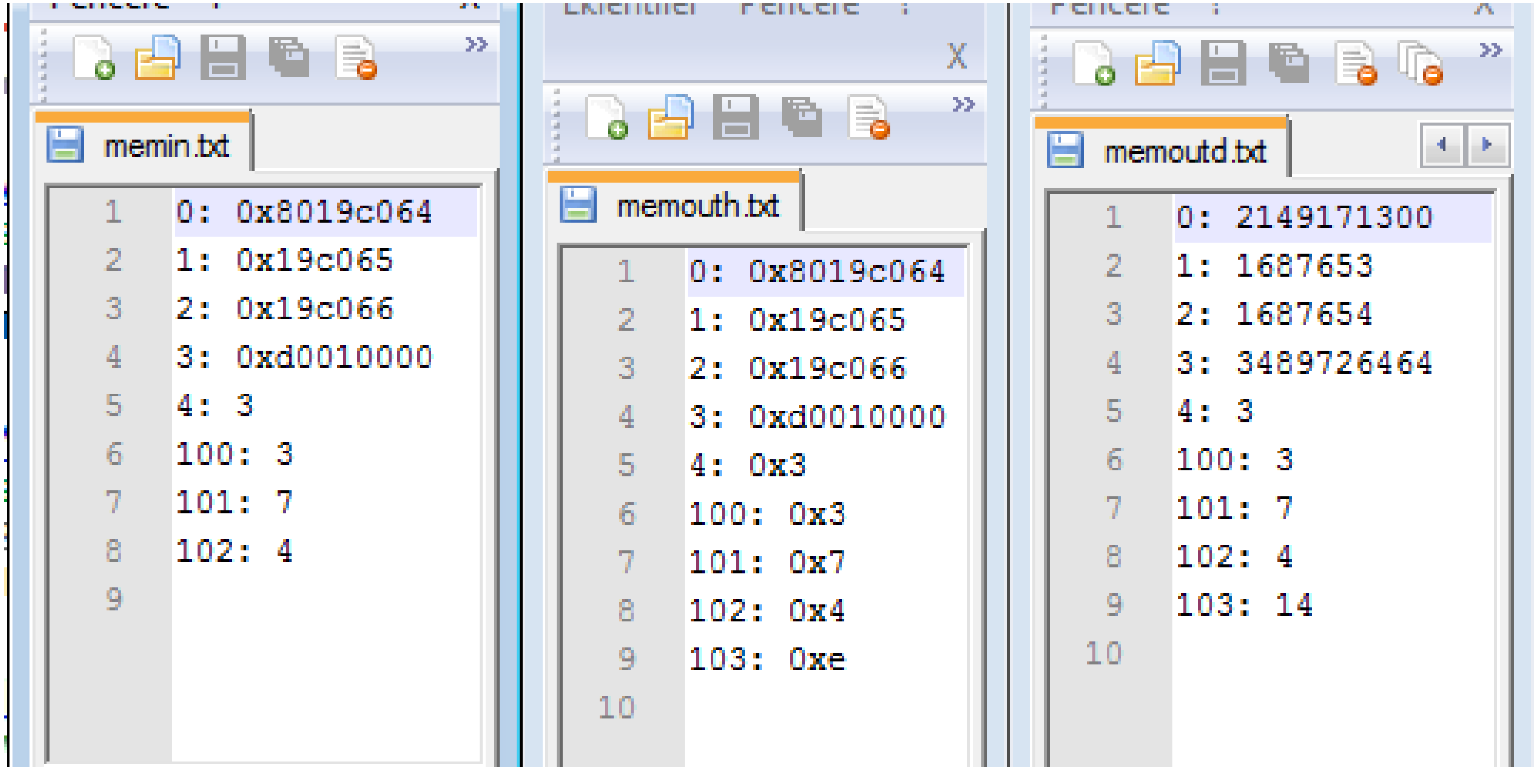
  VerySimpleCPU.exe sample.asm r

- Output of the simulator →



```
D:\PhD\CS240L\Lab4\Sim>SimpleCPU.exe addition.txt r
-------------------
PC: 0
CP 103 100

Before
100: 3
103: 0

After
100: 3
103: 3

-------------------
PC: 1
ADD 103 101

Before
101: 7
103: 3

After
101: 7
103: 10

-------------------
PC: 2
ADD 103 102

Before
102: 4
103: 10

After
102: 4
103: 14

-------------------
PC: 3
BZJi 4 0

Before
4: 3

After
4: 3

>>> exit
exiting...
```

Example: Output Files



```
0: 0x8019c064
1: 0x19c065
2: 0x19c066
3: 0xd0010000
4: 3
100: 3
101: 7
102: 4
```

```
0: 0x8019c064
1: 0x19c065
2: 0x19c066
3: 0xd0010000
4: 0x3
100: 0x3
101: 0x7
102: 0x4
103: 0xe
```

```
0: 2149171300
1: 1687653
2: 1687654
3: 3489726464
4: 3
100: 3
101: 7
102: 4
103: 14
```