Microprocessors
Report of milestone (1)

1) Abanoub Mimi	25-2674
2) Amir George	25-4968
3) Shary George	25-1684
4) Ahmed Elassiuty	25-6759

Brief description of implementation:

The main class in our project is the **Engine** class, responsible for the flow of the simulation. In it we initialize all the other classes, and for the running of the simulation from the top level (calling to execute the instructions, calling to read data and instruction from caches, and calling to write data in caches according to different writing policies). For other classes we have **Cache** class which is the representation of a cache level (both instruction and data). In it the actual reading or writing data into the cache is done.
There is the **Instruction** class which is responsible for executing each instruction according to its type. There is the **Memory** class which is the representation of our 64KB memory. There is the **RegisterFile** which is the representation of our registers. There is the **Parser** class which is responsible for fetching the input to the code and checking it against a set of defined regular expressions. Errors are returned to the user if any are detected.

How work was split:

Abanoub Mimi: I participated in the initial design and skeleton of the project in cooperation with Amir. I was responsible for designing the Engine and Memory classes. I was also responsible for the handing the data cache i.e. loading reading data from caches and writing them according to the write policies.

Amir George: I co-designed of the project with Abanoub. I was responsible for designing the Cache, Instruction and RegisterFile classes. I also implemented the execution part of the 11 instructions.

Shary George: I was responsible for handling the instruction cache, i.e. reading the instructions from caches and loading them according to each cache geometry.

Ahmed Elassiuty: I was responsible for the Parser part i.e. fetching the user input and parsing it according to a set of regular expressions, and returning errors to the user if any.

The 3 Programs:

Note: the user defined input in access memory time is 130 and starting address is 200 (in the 3 programs)

The first program:

```
addi $R1, $R2, 10
addi $R2, $R2, 20
sub $R2, $R2, $R1
beq $R1, $R2, end
addi $R5, $R6,5
end:
sw $R2, $R2, 40
```

This program add value 10 to R1 and 20 to R2 then subtract 10 from R2. Then brach is commited because R1=R2=10 and skip the addi instruction and end withs SW instruction with total 5 instructions executed.

First to setup the memory hierarchy we need to to tell the Simulator the geometry and the cache hit-time in the form of:

1ST hierarchy:

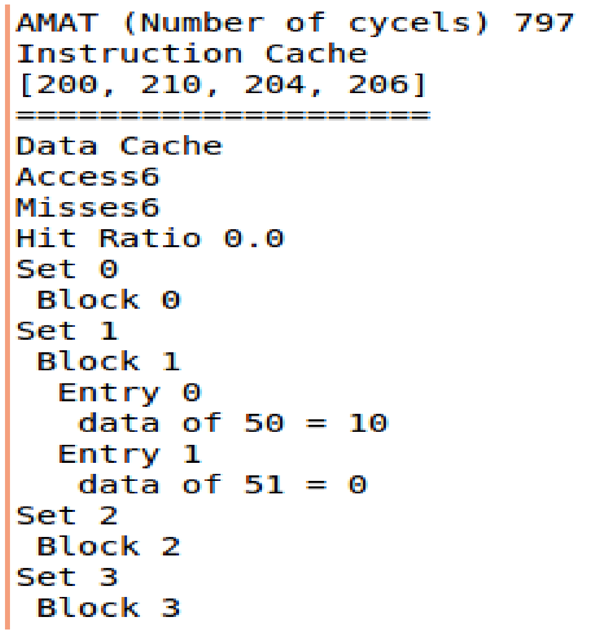
8 2 1 WRITE_BACK WRITE_ALLOCATE 1
16 2 2 WRITE_THROUGH WRITE_AROUND 5

where the input description is `Size of cache, Block size, Associativity, Writing policy in hit, writing policy in miss and number of cycles to access data`

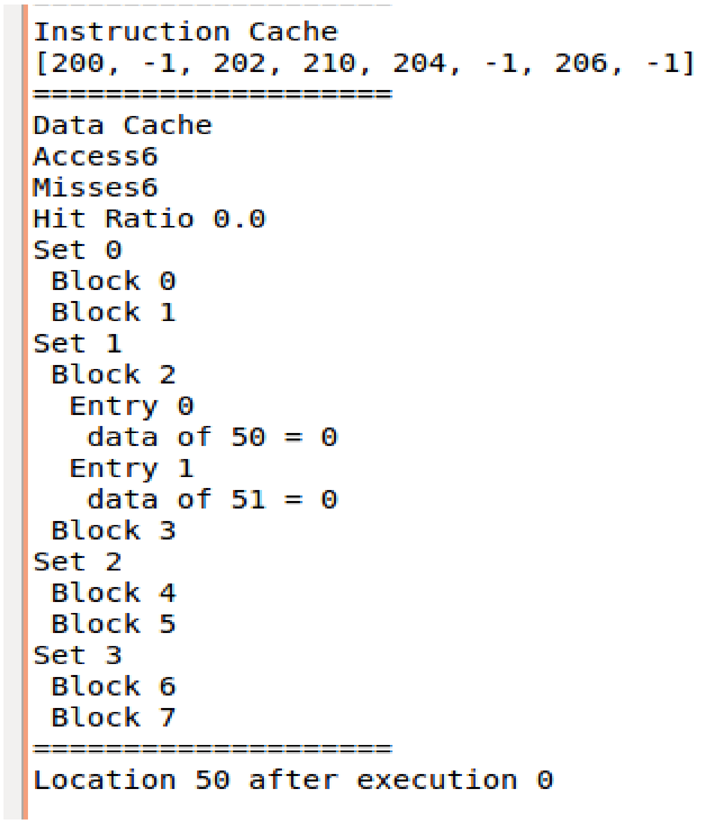
so, this hierarchy represents a direct cache with size 8 and bock size 2 and the writing policies as mentioned in the input.

The result of caches after executing will be

L **1** **L2**



```
AMAT (Number of cycels) 797
Instruction Cache
[200, 210, 204, 206]
=====================
Data Cache
Access6
Misses6
Hit Ratio 0.0
Set 0
 Block 0
Set 1
 Block 1
  Entry 0
   data of 50 = 10
  Entry 1
   data of 51 = 0
Set 2
 Block 2
Set 3
 Block 3
```



```
Instruction Cache
[200, -1, 202, 210, 204, -1, 206, -1]
=====================
Data Cache
Access6
Misses6
Hit Ratio 0.0
Set 0
 Block 0
 Block 1
Set 1
 Block 2
  Entry 0
   data of 50 = 0
  Entry 1
   data of 51 = 0
 Block 3
Set 2
 Block 4
 Block 5
Set 3
 Block 6
 Block 7
=====================
Location 50 after execution 0
```

^{2nd} hierarchy:

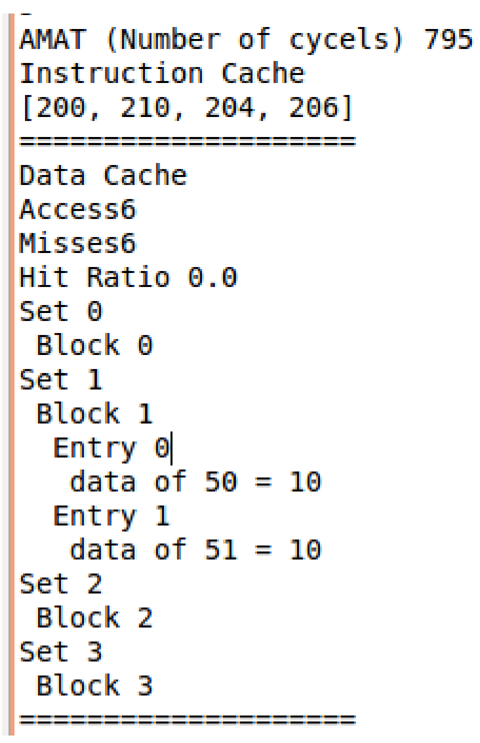
8 2 1 WRITE_THROUGH WRITE_AROUND 5

16 2 2 WRITE_THROUGH WRITE_AROUND 20

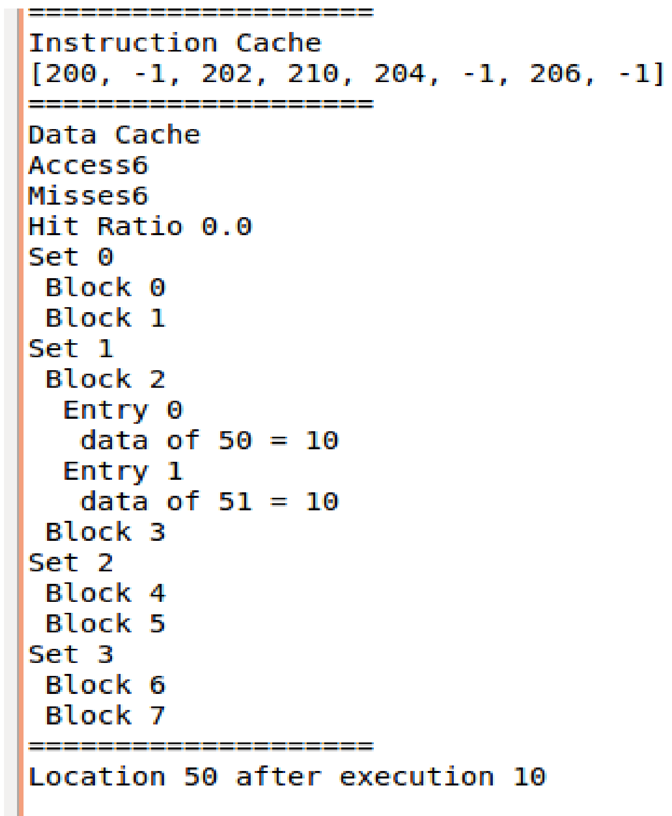
for the 2nd hierarchy we made the writing policies write_through to notice the effect on the memory as each cache will go through the next cache until hit the memory and change the value.
So, the result will be as follows:

L2

L1



```
AMAT (Number of cycels) 795
Instruction Cache
[200, 210, 204, 206]
=====================
Data Cache
Access6
Misses6
Hit Ratio 0.0
Set 0
 Block 0
Set 1
 Block 1
  Entry 0
   data of 50 = 10
  Entry 1
   data of 51 = 10
Set 2
 Block 2
Set 3
 Block 3
=====================
```



```
=====================
Instruction Cache
[200, -1, 202, 210, 204, -1, 206, -1]
=====================
Data Cache
Access6
Misses6
Hit Ratio 0.0
Set 0
 Block 0
 Block 1
Set 1
 Block 2
  Entry 0
   data of 50 = 10
  Entry 1
   data of 51 = 10
 Block 3
Set 2
 Block 4
 Block 5
Set 3
 Block 6
 Block 7
=====================
Location 50 after execution 10
```

The second
program:
// Memory of 20 has data = 10

addi $R2,$R2,10
lw $R1,$R2,10 //r1 = MEM[20] = 10
addi $R3,$R1,10 // r3 =r1+10=20
mult $R5,$R1,$R3 // r5 = r1*r3 **=** 10*20 = 200
sw $R5,$R2,10 // MEM[20]=r5 = 200

1st hierarchy:

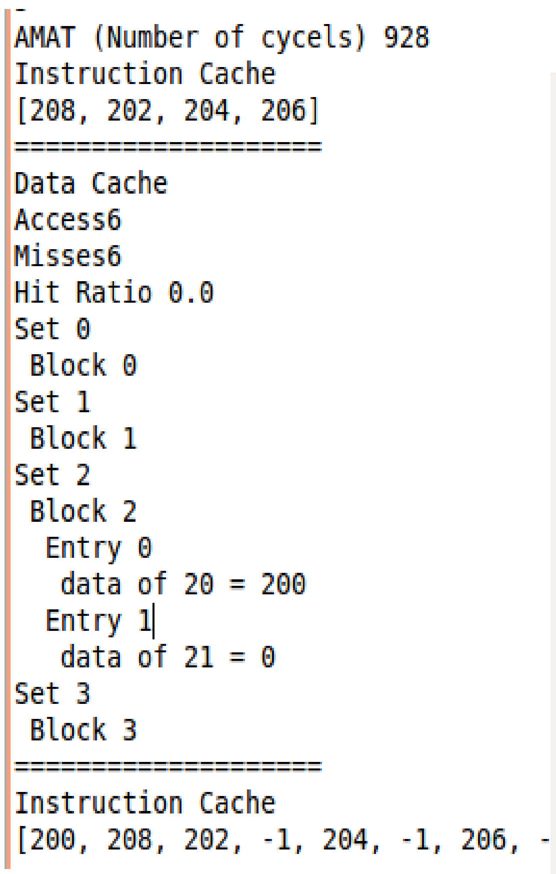
8 2 1 WRITE_BACK WRITE_AROUND 1
16 2 2 WRITE_THROUGH WRITE_AROUND 5

NOTE that 200 are written in the memory

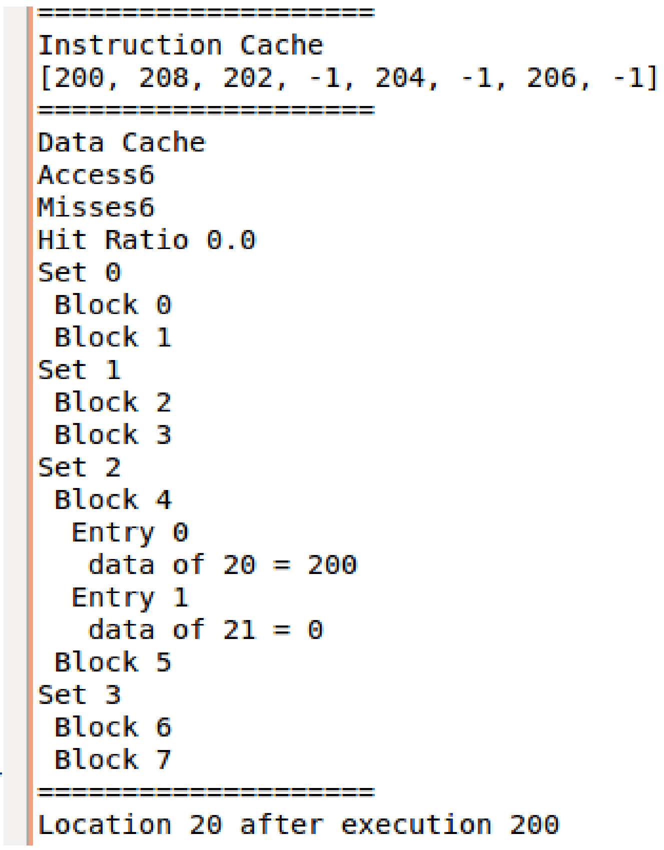
The results are:

L1

L2



```
AMAT (Number of cycels) 928
Instruction Cache
[208, 202, 204, 206]
====================
Data Cache
Access6
Misses6
Hit Ratio 0.0
Set 0
 Block 0
Set 1
 Block 1
Set 2
 Block 2
  Entry 0
   data of 20 = 200
  Entry 1
   data of 21 = 0
Set 3
 Block 3
====================
Instruction Cache
[200, 208, 202, -1, 204, -1, 206, -
```



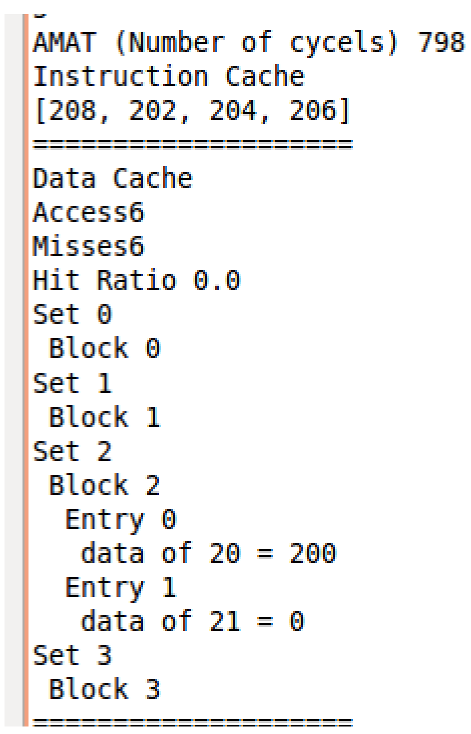
```
====================
Instruction Cache
[200, 208, 202, -1, 204, -1, 206, -1]
====================
Data Cache
Access6
Misses6
Hit Ratio 0.0
Set 0
 Block 0
 Block 1
Set 1
 Block 2
 Block 3
Set 2
 Block 4
  Entry 0
   data of 20 = 200
  Entry 1
   data of 21 = 0
 Block 5
Set 3
 Block 6
 Block 7
====================
Location 20 after execution 200
```

2nd hierarchy:

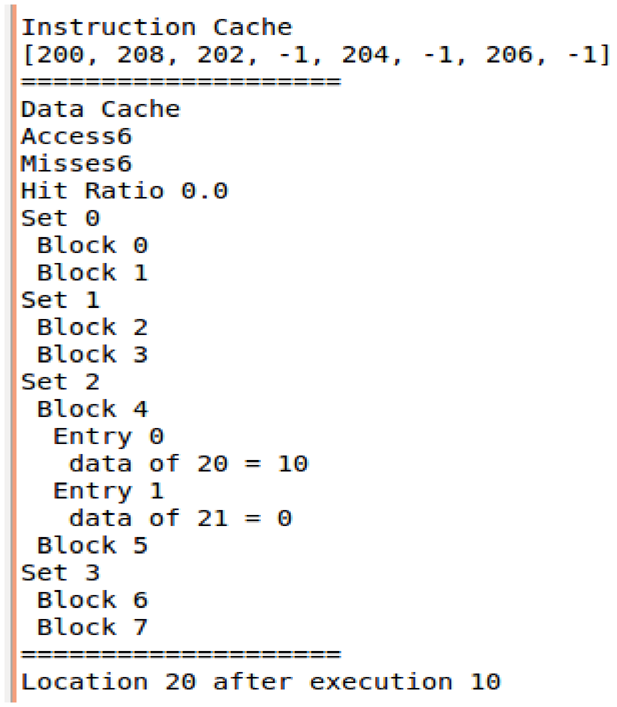
8 2 1 WRITE_THROUGH WRITE_AROUND 5
16 2 2 WRITE_THROUGH WRITE_AROUND 20

Note that 200 are written in the first cache only and the 2nd cache has the old value and marked as DIRTY VALUE!

L1 **L2**



```
AMAT (Number of cycels) 798
Instruction Cache
[208, 202, 204, 206]
====================
Data Cache
Access6
Misses6
Hit Ratio 0.0
Set 0
 Block 0
Set 1
 Block 1
Set 2
 Block 2
  Entry 0
   data of 20 = 200
  Entry 1
   data of 21 = 0
Set 3
 Block 3
====================
```



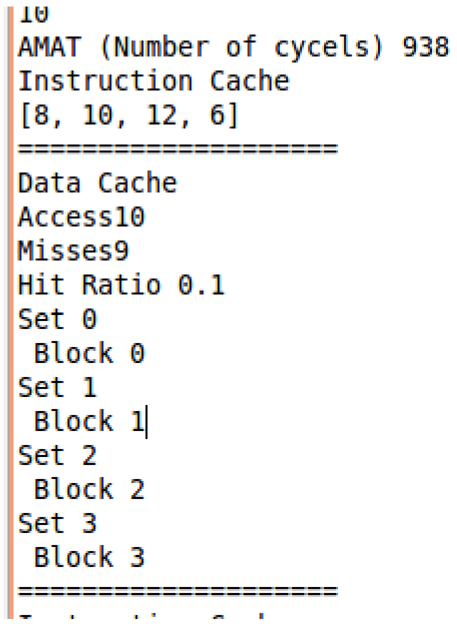
```
Instruction Cache
[200, 208, 202, -1, 204, -1, 206, -1]
====================
Data Cache
Access6
Misses6
Hit Ratio 0.0
Set 0
 Block 0
 Block 1
Set 1
 Block 2
 Block 3
Set 2
 Block 4
  Entry 0
   data of 20 = 10
  Entry 1
   data of 21 = 0
 Block 5
Set 3
 Block 6
 Block 7
====================
Location 20 after execution 10
```

The 3rd program
starting address is 0

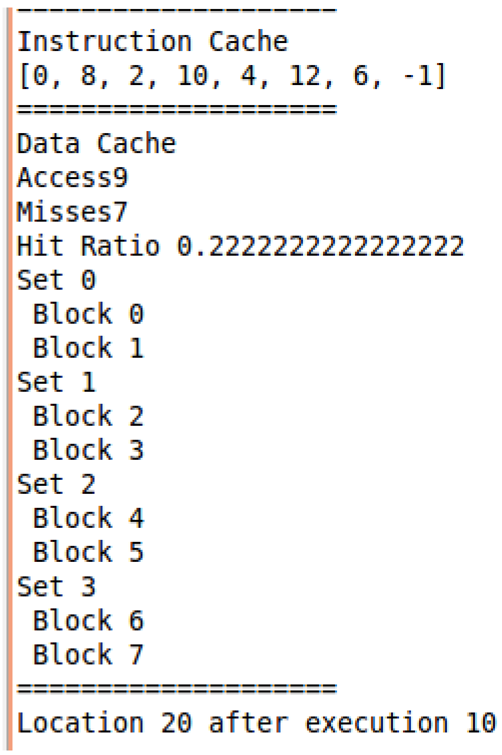
```
200 ADDI R4, R4, #208
202 JALR R3,R4
204 ADDI R5,R5,#23
206 BEQ R5,R5,End
208 AddI R6,R6,#17
Ret R3
End:
ADDI R6,R6,#5
```

1st hierarchy

8 2 1 WRITE_BACK WRITE_AROUND 1
16 2 2 WRITE_THROUGH WRITE_AROUND 5

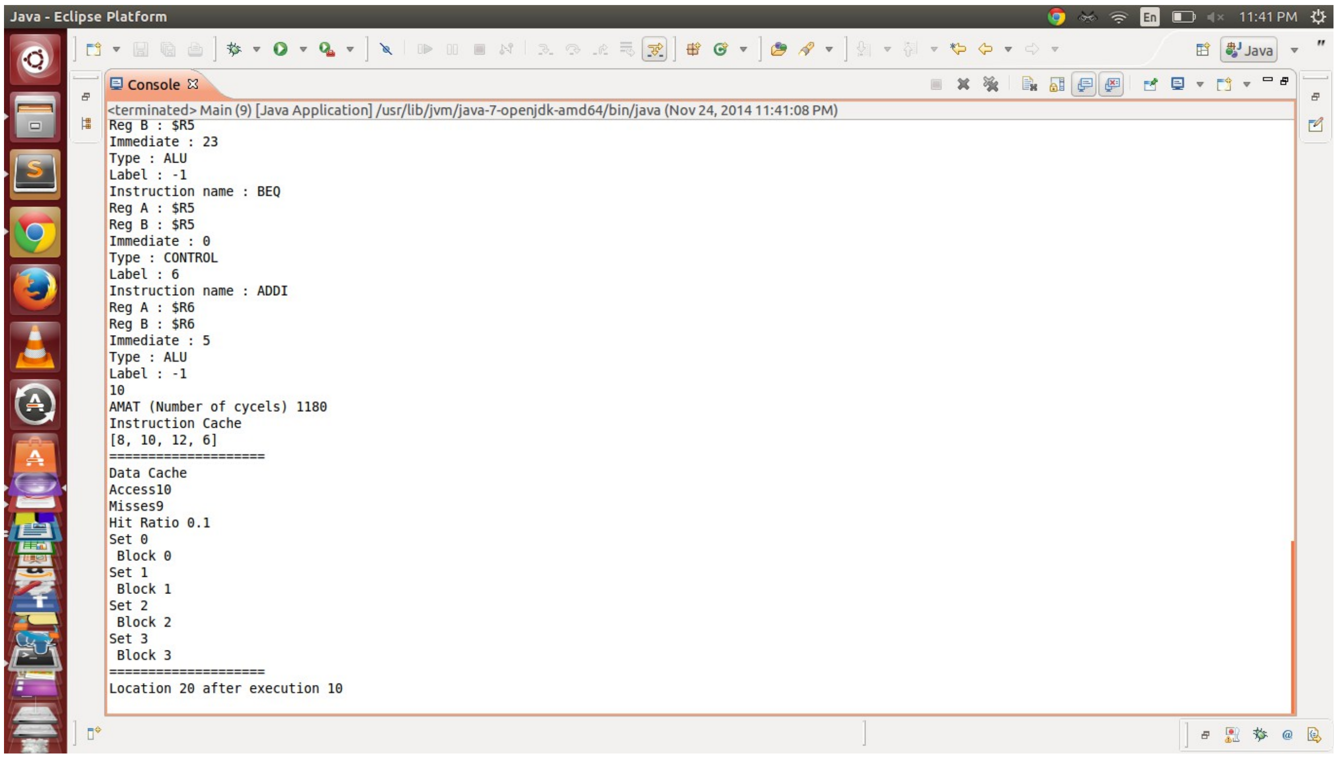


```
10
AMAT (Number of cycels) 938
Instruction Cache
[8, 10, 12, 6]
====================
Data Cache
Access10
Misses9
Hit Ratio 0.1
Set 0
 Block 0
Set 1
 Block 1
Set 2
 Block 2
Set 3
 Block 3
====================
```



```
====================
Instruction Cache
[0, 8, 2, 10, 4, 12, 6, -1]
====================
Data Cache
Access9
Misses7
Hit Ratio 0.2222222222222222
Set 0
 Block 0
 Block 1
Set 1
 Block 2
 Block 3
Set 2
 Block 4
 Block 5
Set 3
 Block 6
 Block 7
====================
Location 20 after execution 10
```

The 2nd hierarchy
8 2 1 WRITE_BACK WRITE_AROUND 1



```
<terminated> Main (9) [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (Nov 24, 2014 11:41:08 PM)
Reg B : $R5
Immediate : 23
Type : ALU
Label : -1
Instruction name : BEQ
Reg A : $R5
Reg B : $R5
Immediate : 0
Type : CONTROL
Label : 6
Instruction name : ADDI
Reg A : $R6
Reg B : $R6
Immediate : 5
Type : ALU
Label : -1
10
AMAT (Number of cycels) 1180
Instruction Cache
[8, 10, 12, 6]
====================
Data Cache
Access10
Misses9
Hit Ratio 0.1
Set 0
 Block 0
Set 1
 Block 1
Set 2
 Block 2
Set 3
 Block 3
====================
Location 20 after execution 10
```