Simulation

The Code

.text

ADDI x2,x0,5

ADDI x3,x0,10

LW x1,x0,10

SUB x4,x3,x2

MUL x5,x2,x3

NAND x6,x4,x2

.data

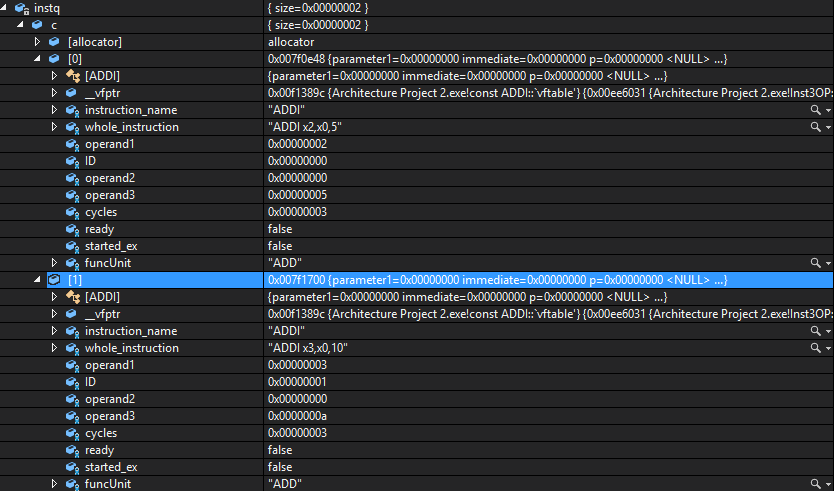
0x00: 20

0x01: 45

0x0A: 22

Cycle 1

Instruction Queue

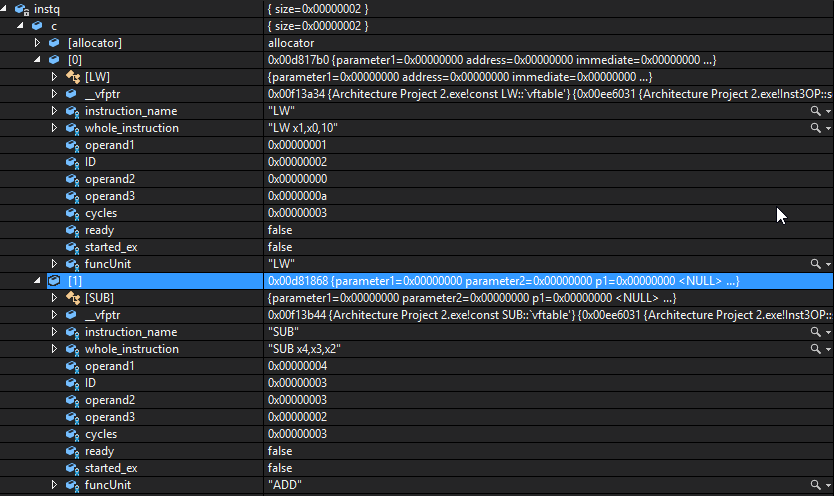


I0 and I1 were fetched from the memory and placed in the instruction queue. These two instructions are ADDI x2,x0,5 and ADDI x3,x0,10. The ROB and the stations are currently empty.

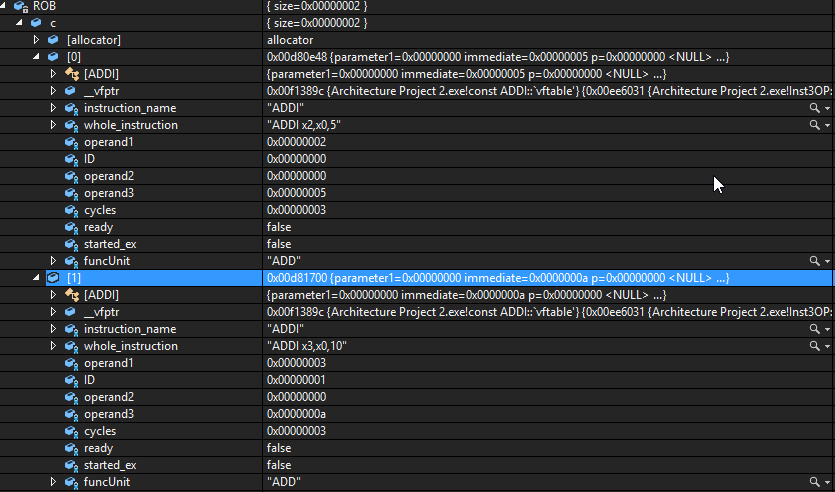
Cycle 2

I2 and I3 are now fetched and placed into the instruction queue. These are LW x1,x0,10 and SUB x4,x3,x2. I0 and I1 were issued and therefore were removed from the instruction queue. They were also placed into the ROB and the ADD stations.

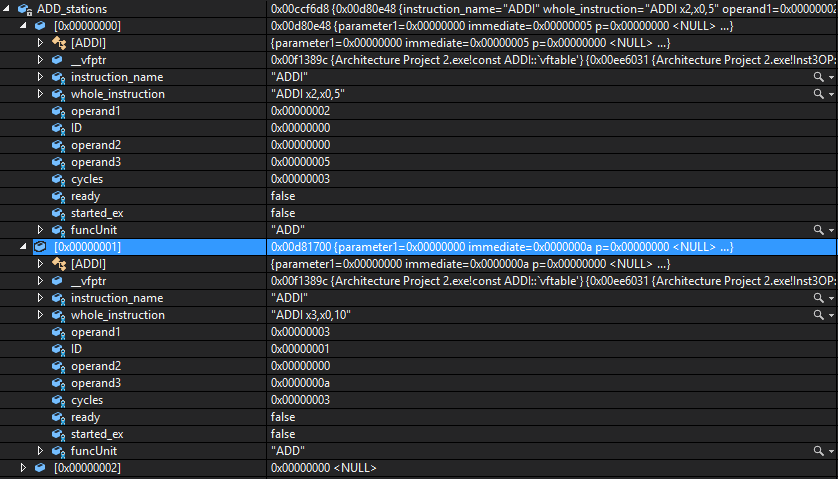
Instruction queue



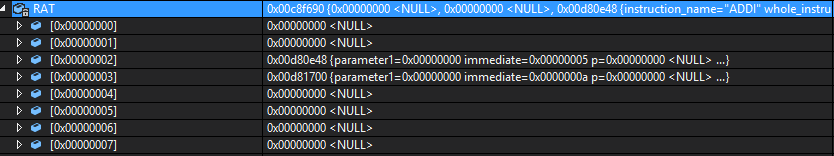
ROB



ADD stations



RAT

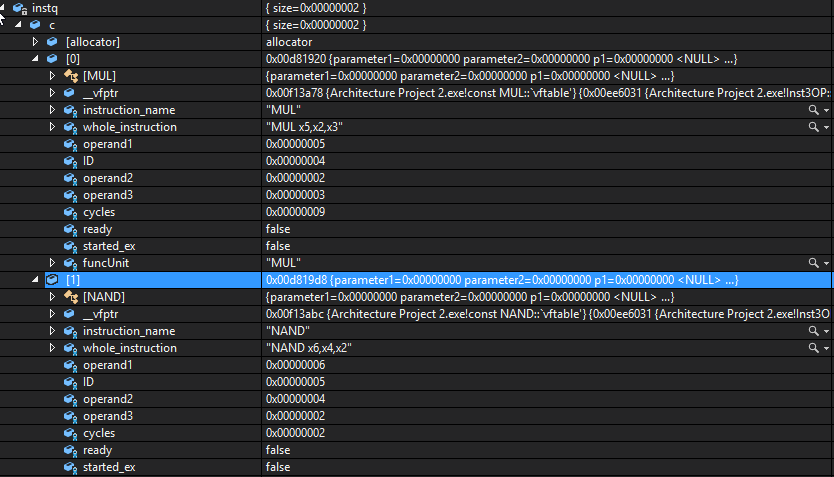


Registers x2 and x3 have been filled in the RAT (Register Alias Table) since these are the destination registers of the first two instructions.

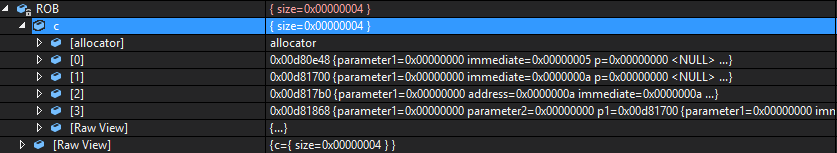
Cycle 3

In this cycle, I0 and I1 start executing. I2 and I3 are issued while I4 and I5 are fetched.

Instruction queue



ROB

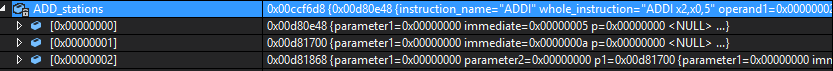


I2 and I3 were added to the ROB when they were issued, meaning the ROB now has 4 entries.

LW station

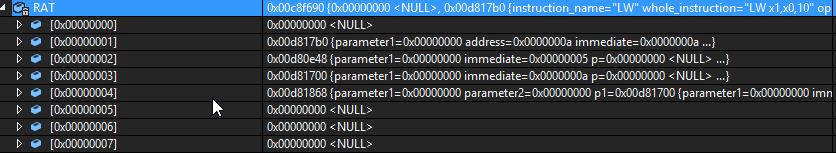


ADD stations



Note that the three add stations are now occupied. This is because I0 and I1 are being executed, while the I3 was just added this cycle.

RAT



Registers x1 and x4 have now been filled, since these are the destination registers for I2 and I3.

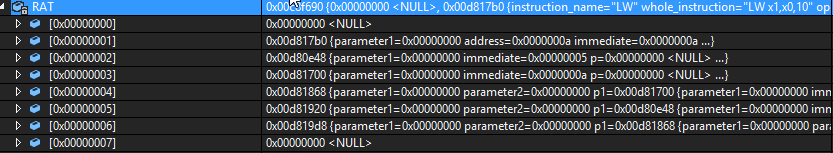
Cycle 4

During this cycle, I0 and I1 are still executing. I2 starts executing. I4 and I5 are issued. Note that I3 cannot start executing during this cycle because register x3 which is produced by I1 is still not available.

Instruction queue:



RAT



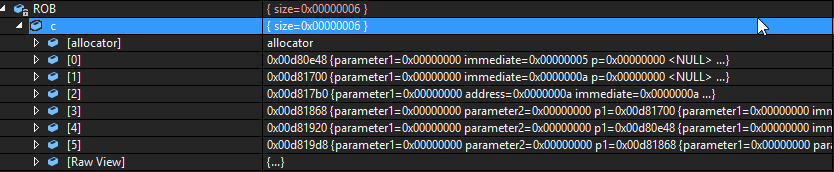
Since all instructions have been issued, it can be seen that the RAT is unchanged, with the instruction queue being empty as well.

MUL & NAND stations



Since I4 and I5 were issued during this cycle, they have been inserted into their own respective stations.

ROB



The ROB now has all 6 instructions, since I4 and I5 were issued during this cycle.

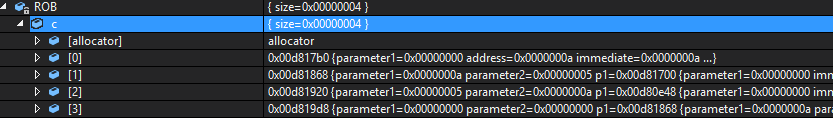
Cycle 5

During this cycle, I0 and I1 finish executing and go on to the writeback phase, which essentially means they will be signaled as ready for committing next cycle. I2 is still executing. I3 can now start executing, since the value from I1 can be forwarded. I4 also starts execution while I5 has to wait for I3, since it produces x4. Therefore, all stations and tables remain the same.

Cycle 6

During this cycle, I0 and I1 are committed, i.e. are written to the register file. I2 finishes execution and is signaled to be ready. I3 and I4 are still being executed while I5 is still waiting for I3 to finish execution.

ROB



The ROB now only has 4 entries, since I0 and I1 have been committed.

LW station



The LW station has now been freed up, since I2 finished execution and went on to the writeback phase.

Cycle 7

I2 is now ready to be committed. I3 finishes execution and passes onto the writeback stage. I4 is still executing. I5 can now start executing, since the value produced by I3 can be forwarded.

ROB



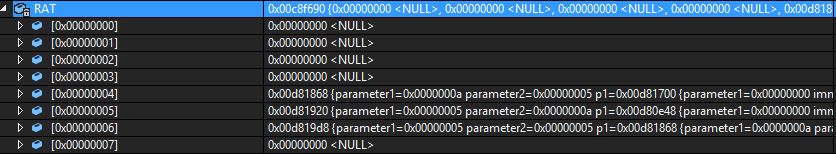
The ROB now has 3 entries only, as I2 is removed since it has committed.

ADD stations



The ADD stations are now all free, since I3 finished execution.

RAT



Register x1 in the RAT has been freed up since I2 has committed.

Cycle 8

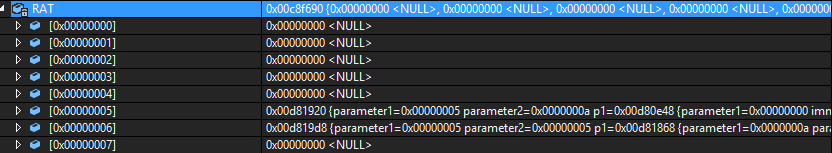
During this cycle, I3 is ready to be committed. I5 finishes execution, since it only takes one cycle. I4 is still being executed.

ROB



Since I3 was committed, it was removed from the ROB.

RAT

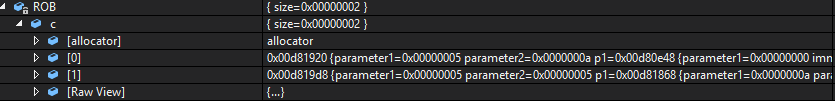


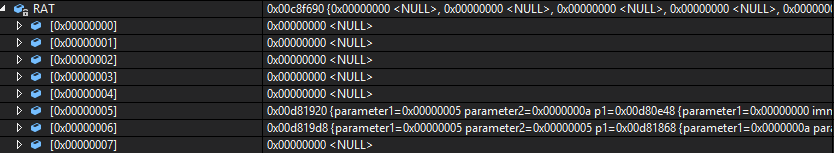
Register x4 was freed up, since I3 committed.

Cycle 9

During this cycle, I5 is ready to be committed. However, since I4 is still executing and committing should be done in order, I5 remains in the ROB until I4 is committed.

ROB & RAT





Cycles 10 to 12

During these cycles, nothing is happening except for the execution of I4.

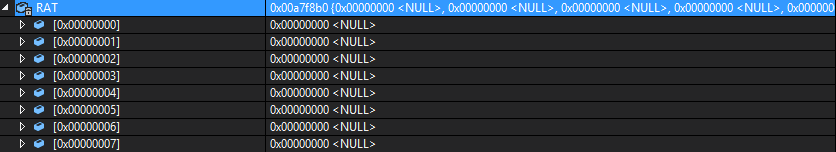
Cycle 13

I4 finishes execution and goes onto the writeback stage.

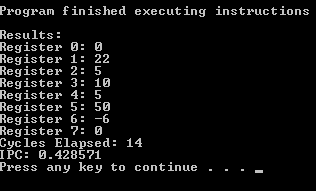
Cycle 14

I4 finally commits, which allows I5 to committed as well. Program stops execution.





Results



Note: The branch miss percentage is not displayed, since there weren’t any branches in the program.