COMPUTER ARCHITECTURE

ASSIGNMENT 2

--------------------------------------------------------------------------------------------------------------------------------------  
ROLL NO : IMT2022067

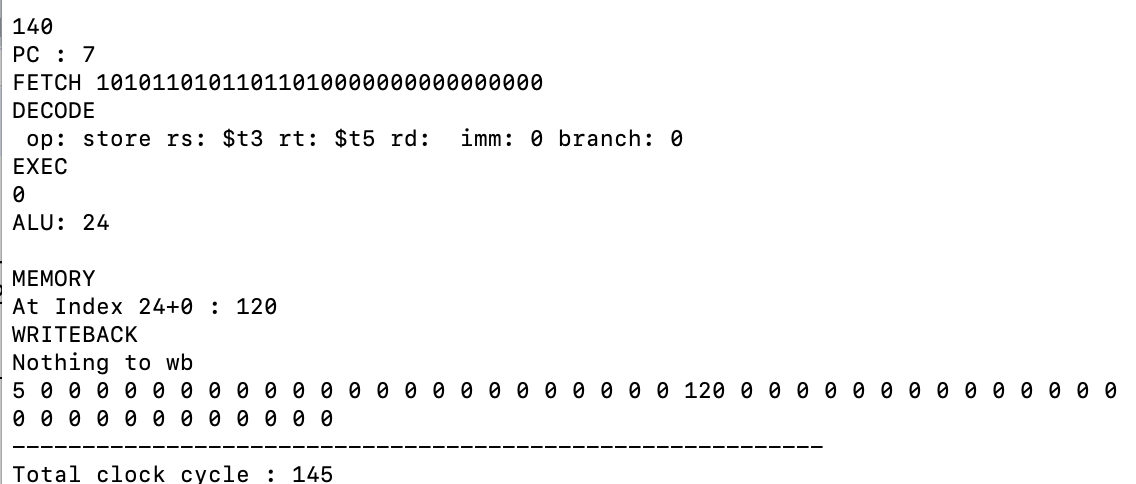
NAME : Aditya Saraf

1)

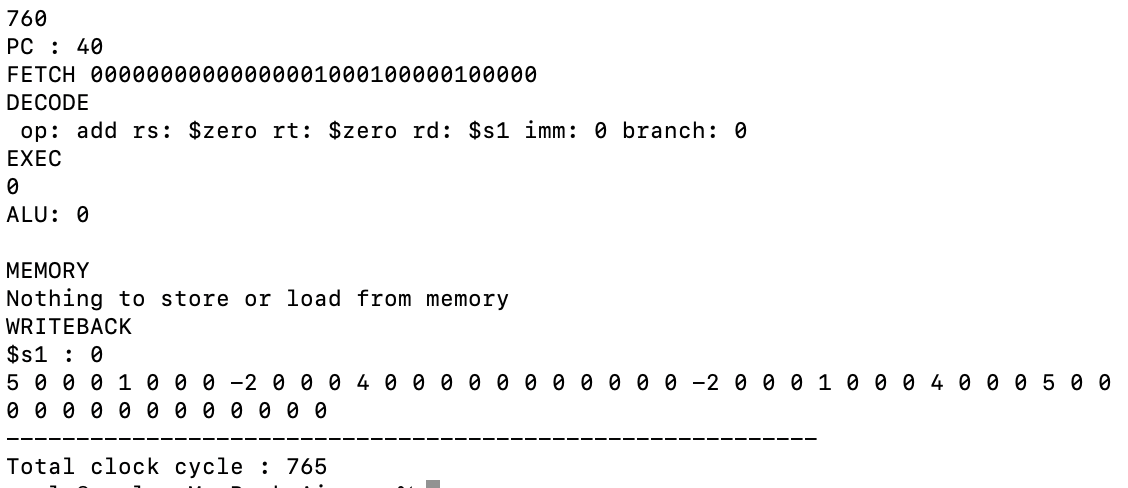
Non-pipelined MIPS processor design

We have tried to implement a non-pipelined MIPS processor design , with minimal functions.

The program which ,we are doing are bubble sorting(which was used in assignment 1) and factorial.



factorial of 5 is stored at index 24, i.e 120



Sorting of 4 integers (left side unsorted(from 0 index) , and right side sorted(from 24index)).

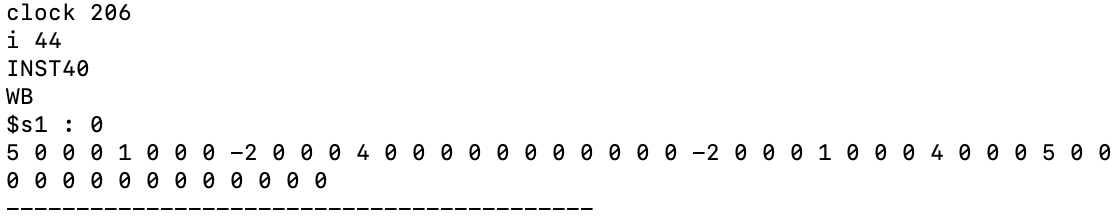
We have tried to make functions for all five stages , and each stage take 1 cycle(assumption).For the memory ,there is inp vector .And we have to manually type the inputs which you want to calculate.We even need to make some changes for $t1(number of integers),$t2(m = input address), $t3(n = output address) registers as its predifined as it was mentioned in assignment 1. Since its a non-pipeline , therefore no stalls and forwarding.Basically , it takes one instruction, then fetch , decode,execute,memory,write back.And does same for all other instruction.

Assumption :

1. It won’t work for other programs , as if there is some functions which I have not used such as j , ja ,bge etc.But, it works for only functions which we have wriiten in the code.
2. We stored all the instructions in a vector ,so that it can be easily iterate or can be jumped for the branch instruction.
3. We are not differentiating the instruction such as into R , J and I type.
4. $at is also used , so that it has to be always zero after each instruction (code is written according to that such that $at remains 0 after instruction).
5. int alu contains the result after EXE,MEM and WB , for after use such as in MEM and WB.

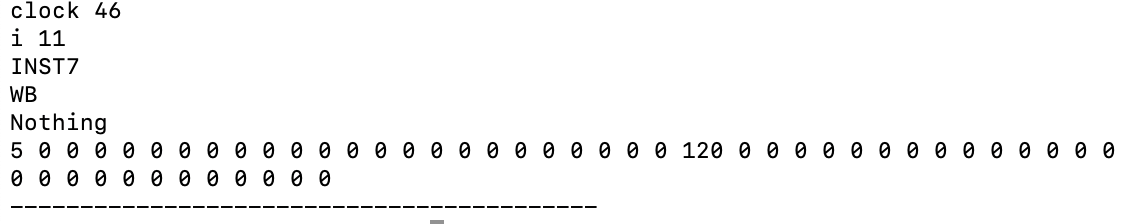
2)

Pipelined MIPS processor design



Same integers as in Non-pipelined Mips

Sorting of 4 integers (left side unsorted(from 0 index) , and right side sorted(from 24index)).



factorial of 5 is stored at index 24, i.e 120

One can compare the clock cycles in non-pipeline and pipeline mips processor.

All the assumptions holds true for pipeline also, which was mentioned in non-pipeline mips.

The code do for stalls and forwarding.(data hazards and branch hazards).

We have used queues as a pipeline registers, to store the previous instructions , so that we can compare if there is a dependency or not.

In branch instruction , we are reading the next instruction .If at the exe stage , branch happens ,then the next two instructions are flushed out and instruction to be fetched is fetched in the next clock cycle.If branch does not happen then the flow continue to happen.

If there is a dependency, then it does forwarding(used cas variable) if possible. Else there is stall (especially for the load operation).

Assumption :

1. Assumed that at stage , it write backs in first half of clock cycle and read in other half.
2. Have rearranged a liitle bit of code so that stalls can be reduced.
3. Have not considered all control signal (if used then , it has been used as a flag or a variable).

Observation :

One can see that clock cycles got reduced by approx 3.5 to 3.7.

More the number of instructions , the clock cycles gets reduced more.

So it majorily takes less time in pipelie mips processor.

-------------------------------------------------THE END ------------------------------------------------------------------------