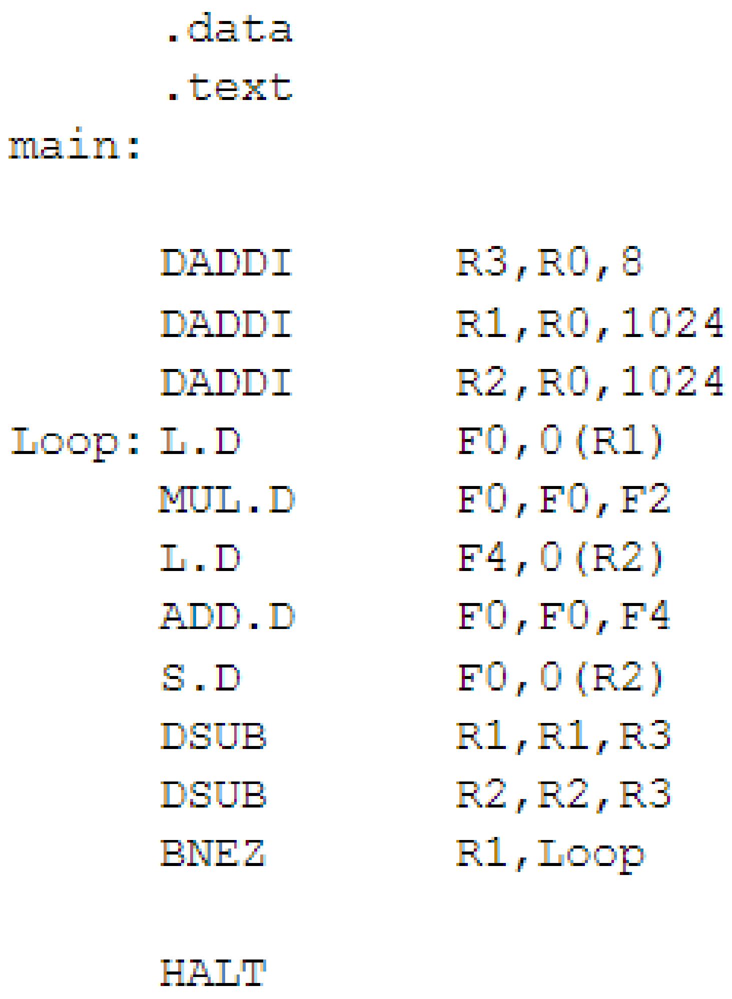
P3 Report

By Keegan Goecke

<u>Initial Code:</u>



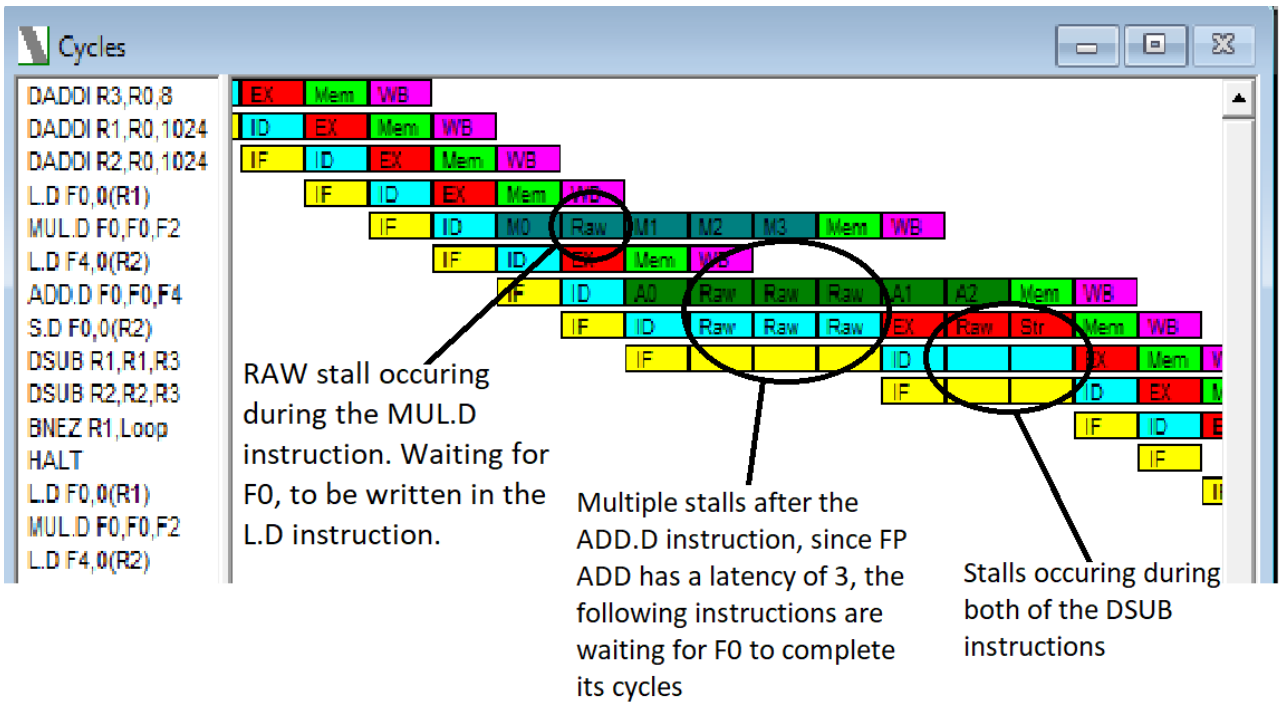
```
        .data
        .text
main:

        DADDI       R3,R0,8
        DADDI       R1,R0,1024
        DADDI       R2,R0,1024
Loop:   L.D         F0,0(R1)
        MUL.D       F0,F0,F2
        L.D         F4,0(R2)
        ADD.D       F0,F0,F4
        S.D         F0,0(R2)
        DSUB        R1,R1,R3
        DSUB        R2,R2,R3
        BNEZ        R1,Loop

        HALT
```

(a) Enable forwarding (check under the Configure tab). Run the code. How many stalls do you see? Can you identify where these stalls occur (the pair of instructions) that cause this stall. Hint: Run in Single Cycle mode using F7. What is the CPI?

Stalls	
RAW	1024
WAW	0
WAR	0
Structual	128
Branch Taken	127
Branch Misprediction	0

Total Stalls: 1279

Execution	
Number of Cycles	1799 Cycles
Number of Instructions	1028 Instructions
Cycles Per Instruction (CPI)	1.750



```
DADDI R3,R0,8
DADDI R1,R0,1024
DADDI R2,R0,1024
L.D F0,0(R1)
MUL.D F0,F0,F2
L.D F4,0(R2)
ADD.D F0,F0,F4
S.D F0,0(R2)
DSUB R1,R1,R3
DSUB R2,R2,R3
BNEZ R1,Loop
HALT
L.D F0,0(R1)
MUL.D F0,F0,F2
L.D F4,0(R2)
```

RAW stall occuring during the MUL.D instruction. Waiting for F0, to be written in the L.D instruction.

Multiple stalls after the ADD.D instruction, since FP ADD has a latency of 3, the following instructions are waiting for F0 to complete its cycles

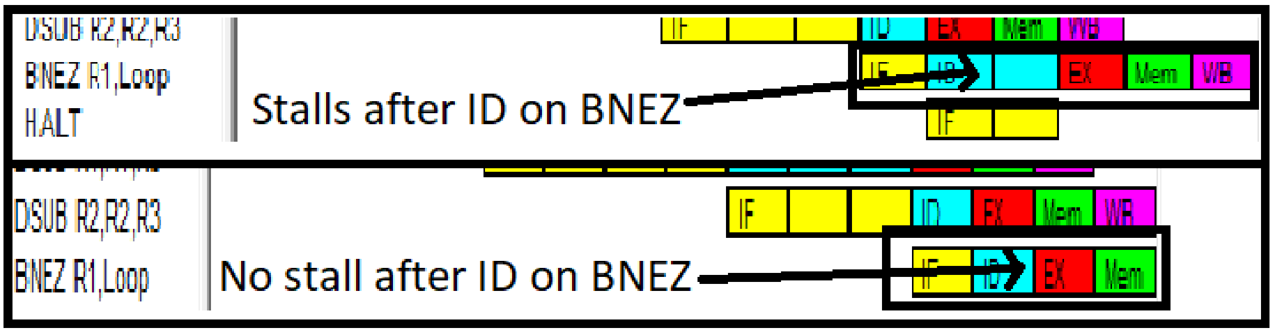
Stalls occuring during both of the DSUB instructions

(b) Execute the code by enabling Enable Branch Target Buffer (check under the Configure tab). How many stalls do you see? How many stalls do you see and what exactly does the Enable Branch Target Buffer do? What is the CPI and what is the speedup when compared to (a)?

Stalls	
RAW	1024
WAW	0
WAR	0
Structual	128
Branch Taken	2
Branch Misprediction	2

Total Stalls: 1156

Enable Branch Target Buffer predicts the address of the target instruction that follows the branch which in return can eliminate stalls as shown below:



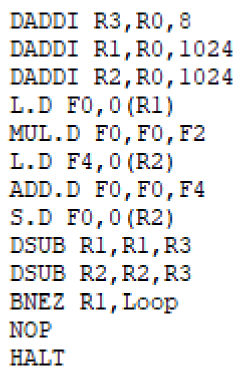
Execution	
Number of Cycles	1676 Cycles
Number of Instructions	1028 Instructions
Cycles Per Instruction (CPI)	1.630

Speedup when compare to (a):

Program 1 = 1.750 * 1028 = 1799
Program 2 = 1.630 * 1028 = 1667

Speedup = 1799 / 1667 = 1.07918

(c) Execute the code by enabling Enable Delay Slot (check under the Configure tab). You will need to put one instruction to be executed, else HALT instruction will stop the code from executing. What is the CPI and the speedup compared to (a). Which scheme is better, branch target buffer or delay slot?



```
DADDI R3,R0,8
DADDI R1,R0,1024
DADDI R2,R0,1024
L.D F0,0(R1)
MUL.D F0,F0,F2
L.D F4,0(R2)
ADD.D F0,F0,F4
S.D F0,0(R2)
DSUB R1,R1,R3
DSUB R2,R2,R3
BNEZ R1,Loop
NOP
HALT
```

Stalls	
RAW	1024
WAW	0
WAR	0
Structual	128
Branch Taken	0
Branch Misprediction	0

Total Stalls: 1152

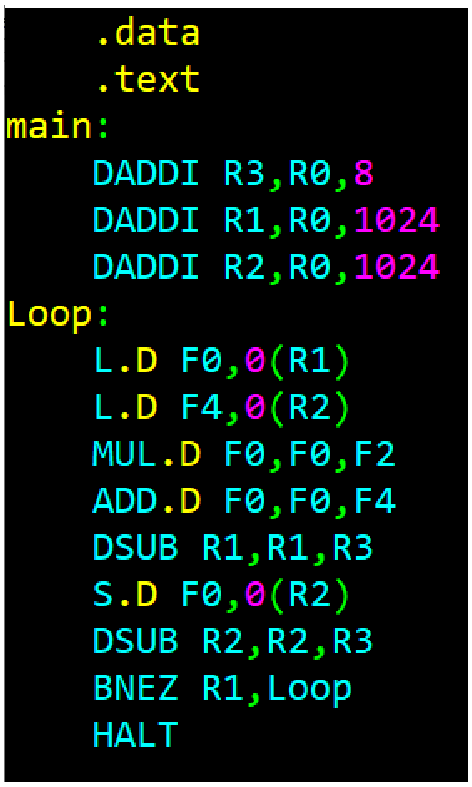
Execution	
Number of Cycles	1800 Cycles
Number of Instructions	1156 Instructions
Cycles Per Instruction (CPI)	1.557

Speedup = 1799/1800 = .999444

Both the CPI and Speedup compared to (a) are worse, it is better to use BTB in this example.

(d) Re-arrange the loop without unrolling. You can move individual instructions, however the output of this dummy loop should be exactly the same i.e. adjust the offset for memory instructions (load/store). Can you reduce the stalls for this code? What is the new CPI and the speedup when compared to (a)?

Re-arranged loop:



```
    .data
    .text
main:
    DADDI R3,R0,8
    DADDI R1,R0,1024
    DADDI R2,R0,1024
Loop:
    L.D F0,0(R1)
    L.D F4,0(R2)
    MUL.D F0,F0,F2
    ADD.D F0,F0,F4
    DSUB R1,R1,R3
    S.D F0,0(R2)
    DSUB R2,R2,R3
    BNEZ R1,Loop
    HALT
```

Stalls	
RAW	768
WAW	0
WAR	0
Structual	128
Branch Taken	127
Branch Misprediction	0

Total Stalls: 1023 (original was 1279)
Stalls reduced by: 1279 – 1023 = 256

Execution	
Number of Cycles	1671 Cycles
Number of Instructions	1028 Instructions
Cycles Per Instruction (CPI)	1.625

Speedup = 1799 / 1671 = 1.06463

(e) Now, transform the loop by unrolling the loop, reschedule the instructions, enable delay slot or branch target buffer to completely minimize the stalls. What is the CPI and what is the speedup when compared to (a)?

Unrolled the loop 4 times with re-arranged instructions as well as BTB enabled:

```
DADDI R3,R0,8
DADDI R1,R0,1024
DADDI R2,R0,1024
L.D F0,0(R1)
L.D F4,0(R2)
MUL.D F0,F0,F2
ADD.D F0,F0,F4
DSUB R1,R1,R3
S.D F0,0(R2)
DSUB R2,R2,R3
L.D F0,0(R1)
L.D F4,0(R2)
MUL.D F0,F0,F2
ADD.D F0,F0,F4
DSUB R1,R1,R3
S.D F0,0(R2)
DSUB R2,R2,R3
L.D F0,0(R1)
L.D F4,0(R2)
MUL.D F0,F0,F2
ADD.D F0,F0,F4
DSUB R1,R1,R3
S.D F0,0(R2)
DSUB R2,R2,R3
L.D F0,0(R1)
L.D F4,0(R2)
MUL.D F0,F0,F2
ADD.D F0,F0,F4
DSUB R1,R1,R3
S.D F0,0(R2)
DSUB R2,R2,R3
BNEZ R1,Loop
HALT
```

Stalls	
RAW	768
WAW	0
WAR	0
Structual	128
Branch Taken	2
Branch Misprediction	2

Total Stalls: 900

Execution	
Number of Cycles	1452 Cycles
Number of Instructions	932 Instructions
Cycles Per Instruction (CPI)	1.558

Speedup = 1799/1452 = 1.23898