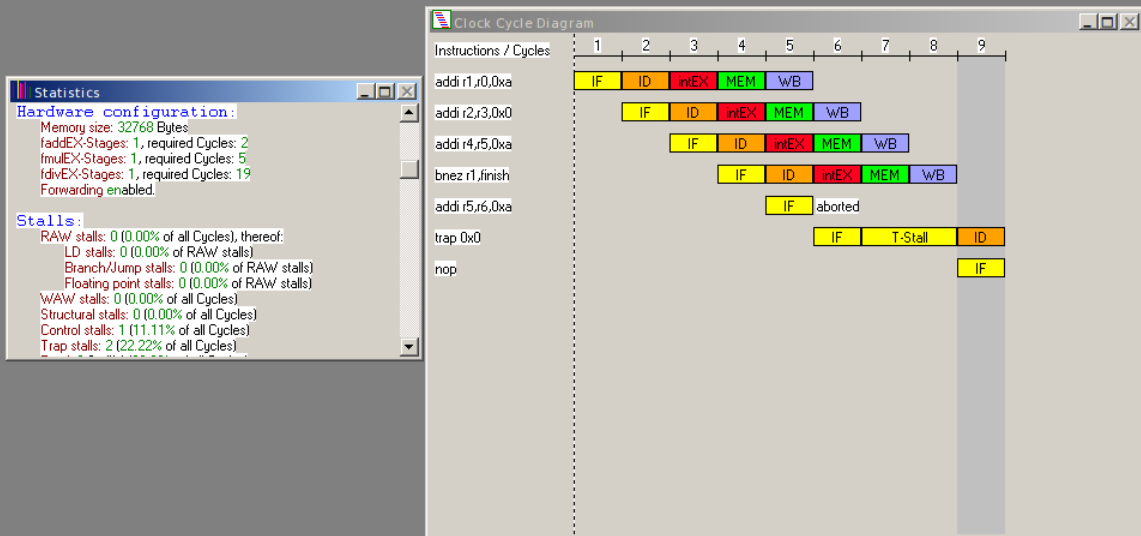


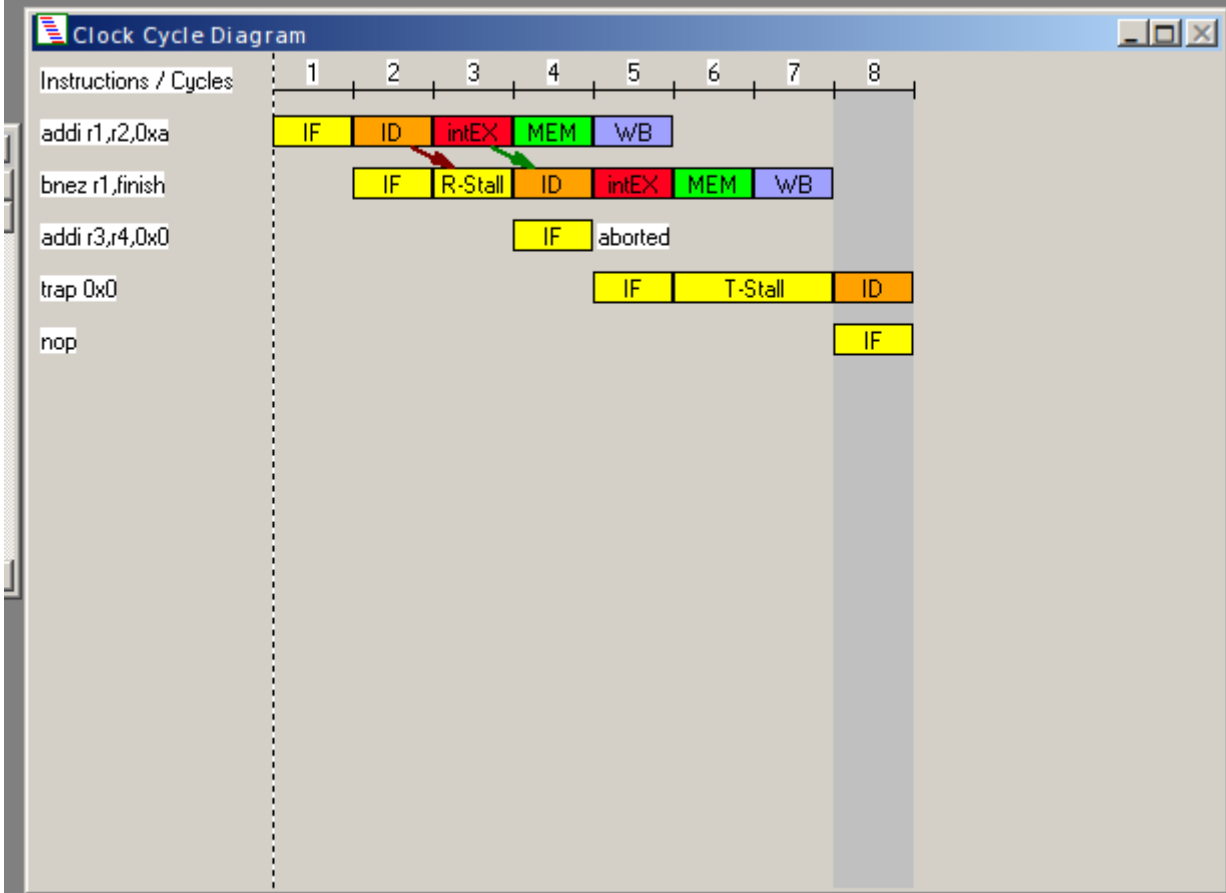
0a)

Control stall:



0b)

Stall in ID stage:



1a)

With data forwarding enabled: 159 cycles

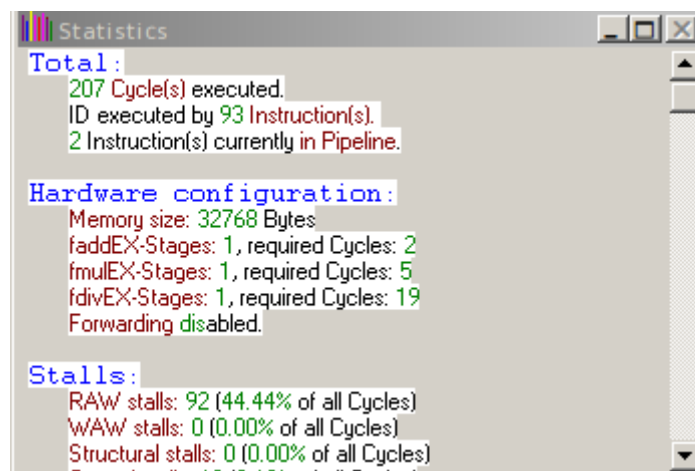


```
Statistics
Total:
159 Cycle(s) executed.
ID executed by 93 Instruction(s).
2 Instruction(s) currently in Pipeline.

Hardware configuration:
Memory size: 32768 Bytes
faddEX-Stages: 1, required Cycles: 2
fmulEX-Stages: 1, required Cycles: 5
fdivEX-Stages: 1, required Cycles: 19
Forwarding enabled.

Stalls:
RAW stalls: 44 (27.67% of all Cycles), thereof:
LD stalls: 13 (29.54% of RAW stalls)
Branch/Jump stalls: 1 (2.27% of RAW stalls)
```

With data forwarding disabled: 207 cycles



```
Statistics
Total:
207 Cycle(s) executed.
ID executed by 93 Instruction(s).
2 Instruction(s) currently in Pipeline.

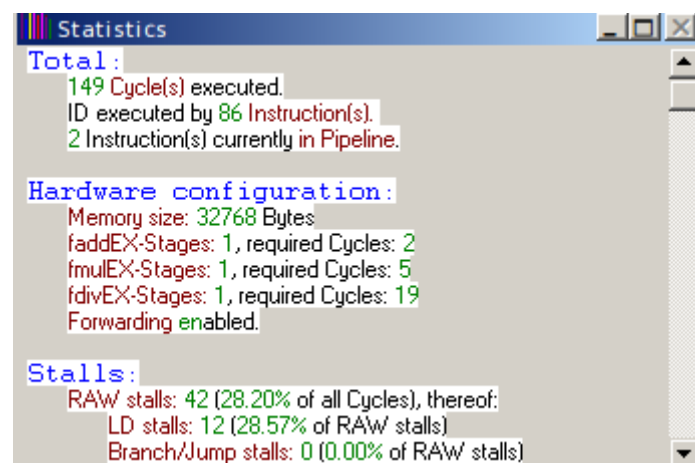
Hardware configuration:
Memory size: 32768 Bytes
faddEX-Stages: 1, required Cycles: 2
fmulEX-Stages: 1, required Cycles: 5
fdivEX-Stages: 1, required Cycles: 19
Forwarding disabled.

Stalls:
RAW stalls: 92 (44.44% of all Cycles)
WAW stalls: 0 (0.00% of all Cycles)
Structural stalls: 0 (0.00% of all Cycles)
```

Speedup with forwarding = (Number of instructions with forwarding)/(Number of instructions with forwarding)
= 207/159 = 1.31

1b)

With forwarding enabled and only one branch instruction: 149 cycles



```
Statistics
Total:
149 Cycle(s) executed.
ID executed by 86 Instruction(s).
2 Instruction(s) currently in Pipeline.

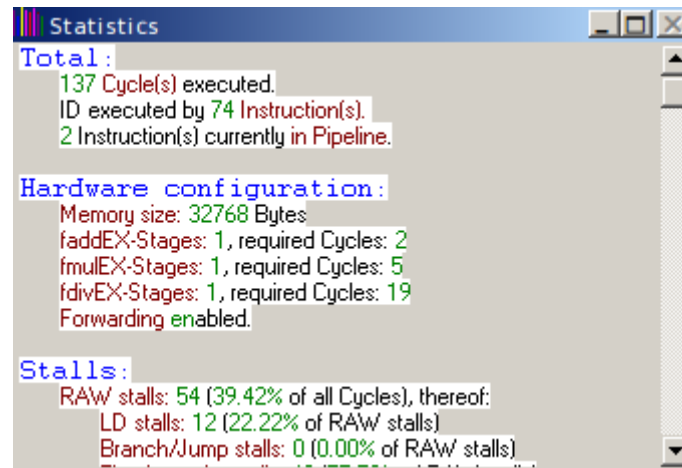
Hardware configuration:
Memory size: 32768 Bytes
faddEX-Stages: 1, required Cycles: 2
fmulEX-Stages: 1, required Cycles: 5
fdivEX-Stages: 1, required Cycles: 19
Forwarding enabled.

Stalls:
RAW stalls: 42 (28.20% of all Cycles), thereof:
LD stalls: 12 (28.57% of RAW stalls)
Branch/Jump stalls: 0 (0.00% of RAW stalls)
```

Speedup with one branch instructions and forwarding = $159/149 = 1.07$

2a)

With inline function, number of cycles = 137



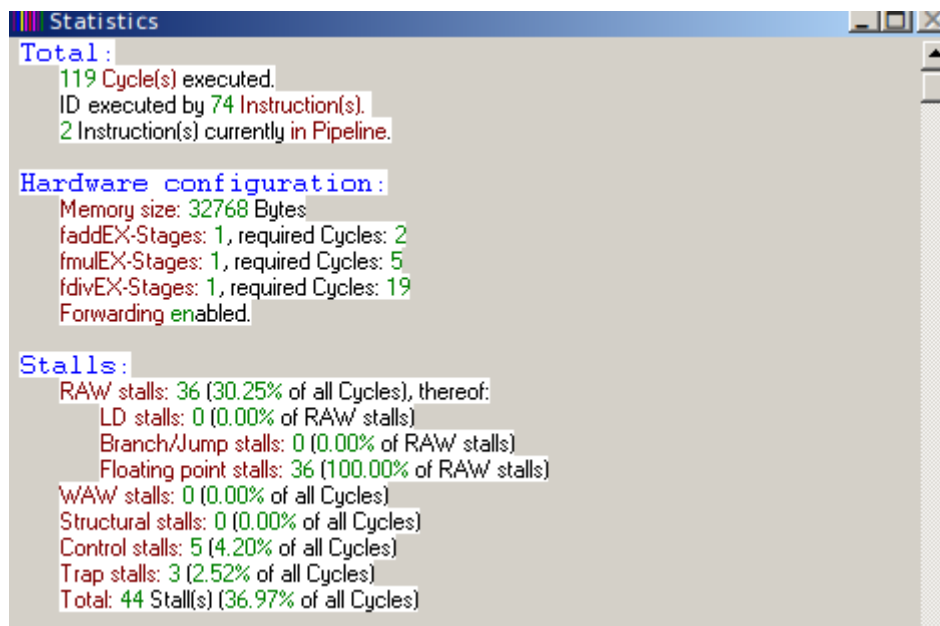
```
Statistics
Total:
137 Cycle(s) executed.
ID executed by 74 Instruction(s).
2 Instruction(s) currently in Pipeline.

Hardware configuration:
Memory size: 32768 Bytes
faddEX-Stages: 1, required Cycles: 2
fmulEX-Stages: 1, required Cycles: 5
fddivEX-Stages: 1, required Cycles: 19
Forwarding enabled.

Stalls:
RAW stalls: 54 (39.42% of all Cycles), thereof:
LD stalls: 12 (22.22% of RAW stalls)
Branch/Jump stalls: 0 (0.00% of RAW stalls)
```

3a)

After scheduling the code properly, Number of cycles: 119



```
Statistics
Total:
119 Cycle(s) executed.
ID executed by 74 Instruction(s).
2 Instruction(s) currently in Pipeline.

Hardware configuration:
Memory size: 32768 Bytes
faddEX-Stages: 1, required Cycles: 2
fmulEX-Stages: 1, required Cycles: 5
fddivEX-Stages: 1, required Cycles: 19
Forwarding enabled.

Stalls:
RAW stalls: 36 (30.25% of all Cycles), thereof:
LD stalls: 0 (0.00% of RAW stalls)
Branch/Jump stalls: 0 (0.00% of RAW stalls)
Floating point stalls: 36 (100.00% of RAW stalls)
WAW stalls: 0 (0.00% of all Cycles)
Structural stalls: 0 (0.00% of all Cycles)
Control stalls: 5 (4.20% of all Cycles)
Trap stalls: 3 (2.52% of all Cycles)
Total: 44 Stall(s) (36.97% of all Cycles)
```

Speedup after scheduling = $137/119 = 1.15$

Changes made:

- 1) Pushed mult instruction down by a step - This lets loading values into register for mult ($a[i]*b[i]$) to finish decreasing stalls
- 2) Loaded r10 three steps before it was needed – This helps in pushing mult instruction down as well as avoiding stall for r10 to be loaded
- 3) Scheduled loading $d[i]$ before it was needed – This pushes multiplication instruction of inline function further down so that calculation of $a[i]*b[i]$ is finished before multiplying it by alpha
- 4) Delayed adding $d[i]$ to result of inline function – This prevents stalls while waiting for mult to execute

5) delayed storing value to d[i] by moving increment of i up – This removes stall while waiting for add to execute

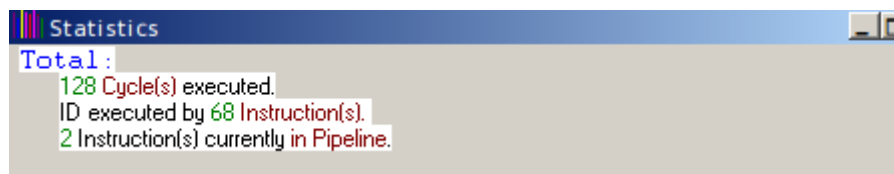
3b)

Couldn't completely remove stall (but decreased by 1 stall) while calculating d[i] from mult instruction of inline function due to it taking more cycles

Couldn't completely remove stall (but decreased by 1 stall) while calculating a[i]*b[i] from mult instruction due to it taking more cycles

4)

Number of cycles after unrolling = 128



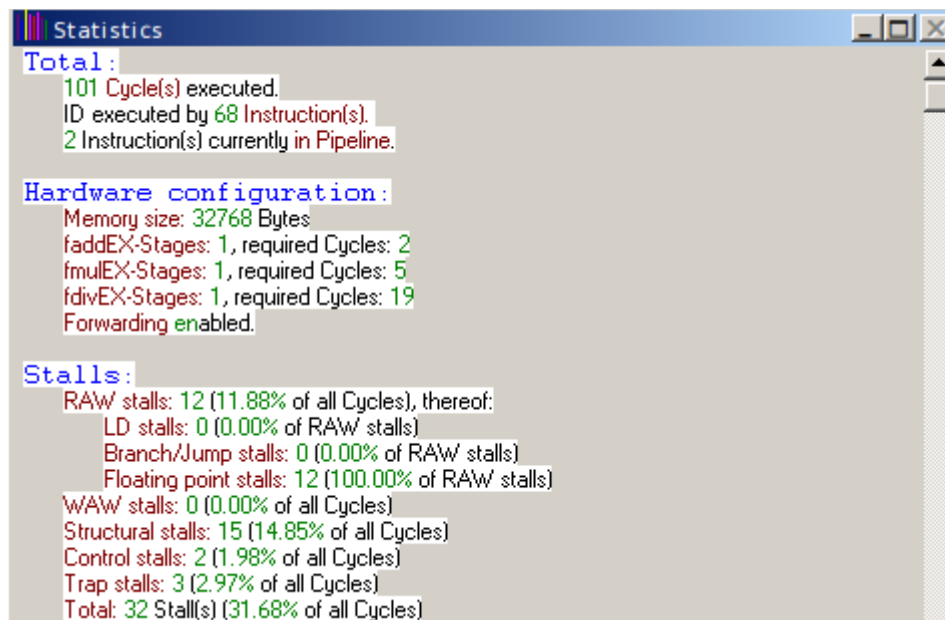
The number of cycles decreased by 9 in comparison to 2-inline.

This is due to number of loops decreasing from 6 to 3 which means 3 branch instructions are removed which lead to $3 \times 2 = 6$ cycle drop (2 cycles per branch one due to instruction itself and another due to nop that will be put after branch by DLX)

Another 3 cycles are reduced due to decrementing the counter by 2 at a time instead of 1

5a)

Number of cycles after unrolling and scheduling = 101



Speedup = $128/101 = 1.27$

Clearly speedup ratio with unrolling and scheduling(1.27) is greater than speedup with just scheduling(1.15)

Modifications:

The code was modified by interleaving the instructions of two iterations that were unrolled
The registers were renamed so as not to conflict between instructions.

This interleaving provided enough gap between two instructions of each iteration so that stalls can be reduced

5b)

But this unrolling and interleaving introduced structural stalls because multiplication unit takes up 5 cycles for execution step and for the next 5 cycles another mult instruction cannot be executed. This causes some stalls. These kinds of stalls are the only ones left that cannot be masked by scheduling and unrolling for the given problem