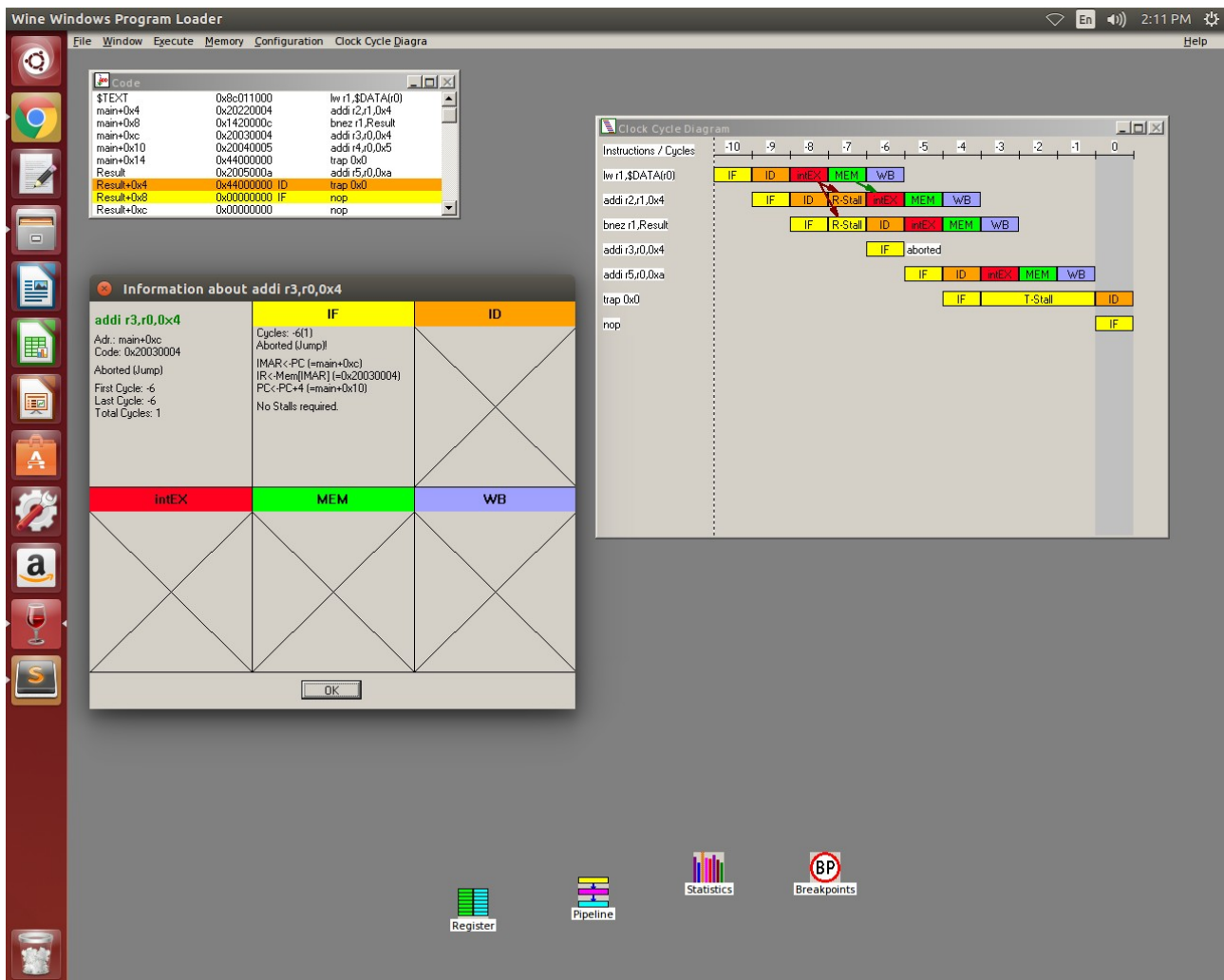


140050002
Question 0a

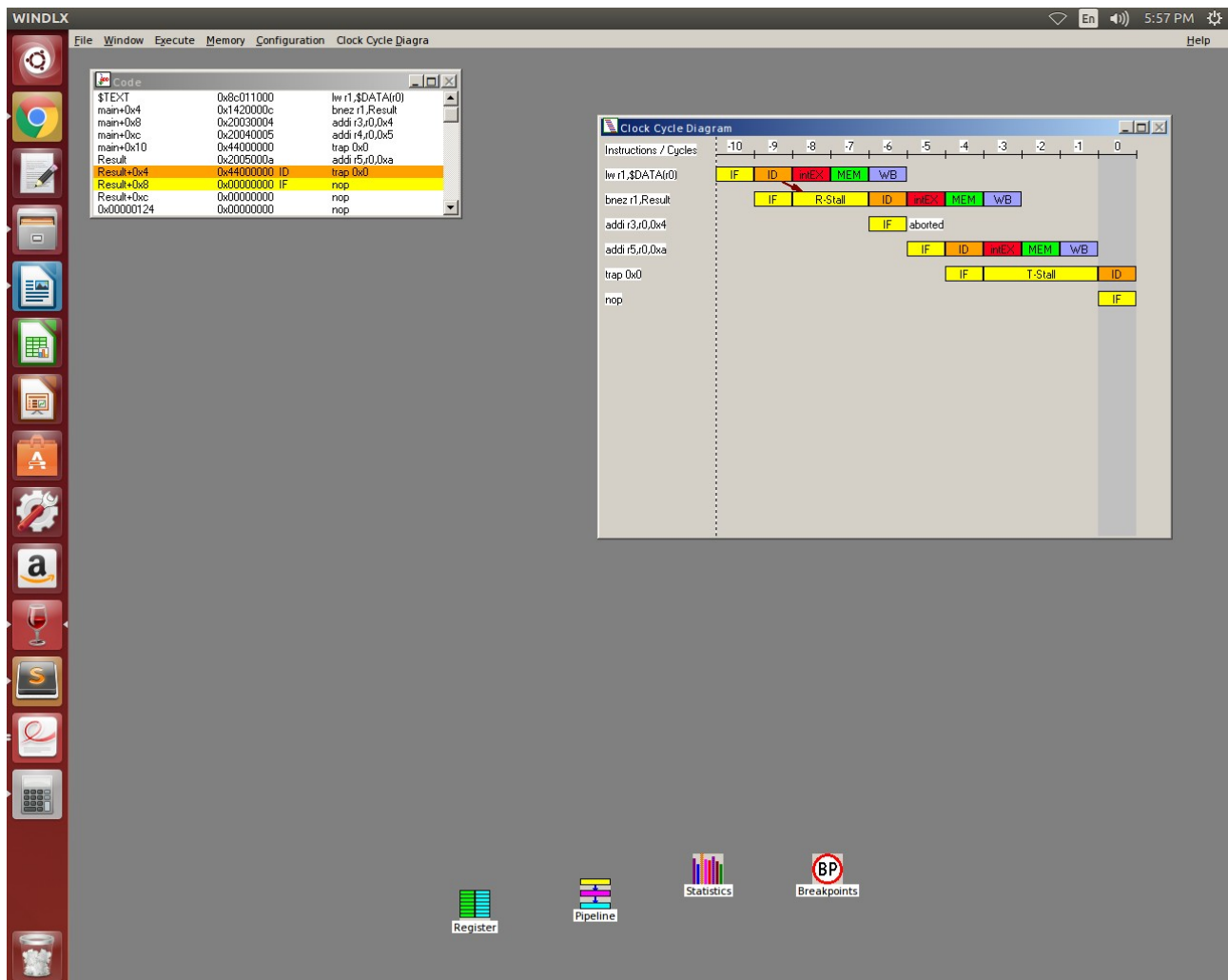


Here because of branch add instruction has been aborted.
Due to static prediction, instruction after bnez is executed but due to branching, previously started instruction must be aborted.

Question 0b

No such program is possible. Stall in ID in branch occurs only when values required for execution is not available. But branch doesn't do any execution in IntEx stage, stall is not possible.

However stall in IF stage possible, because required registers are not available.



Question 1a

Number of clock cycle with forwarding : 203

Number of clock cycle without forwarding : 295

Speed up : $295 / 203 = 1.45$

With forwarding :

```
X - [Statistics]
File Window Execute Memory Configuration Statis
Total:
203 Cycle(s) executed.
ID executed by 132 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
fdvEX-Stages: 1, required Cycles: 19
Forwarding enabled.

Stalls:
RAW stalls: 49 (24.14% of all Cycles), thereof:
  LD stalls: 12 (24.50% of RAW stalls)
  Branch/Jump stalls: 7 (14.28% of RAW stalls)
  Floating point stalls: 30 (61.22% of RAW stalls)
WAW stalls: 0 (0.00% of all Cycles)
Structural stalls: 0 (0.00% of all Cycles)
Control stalls: 19 (9.36% of all Cycles)
Trap stalls: 2 (0.98% of all Cycles)
Total: 70 Stall(s) (34.48% of all Cycles)

Conditional Branches):
Total: 13 (9.85% of all Instructions), thereof:
  taken: 7 (53.85% of all cond. Branches)
  not taken: 6 (46.15% of all cond. Branches)

Load-/Store-Instructions:
Total: 37 (28.03% of all Instructions), thereof:
  Loads: 25 (67.57% of Load-/Store-Instructions)
  Stores: 12 (32.43% of Load-/Store-Instructions)

Floating point stage instructions:
Total: 12 (9.09% of all Instructions), thereof:
  Additions: 0 (0.00% of Floating point stage inst.)
  Multiplications: 12 (100.00% of Floating point stage inst.)
  Divisions: 0 (0.00% of Floating point stage inst.)

Traps:
Traps: 1 (0.76% of all Instructions)
```

Without forwarding:

```
X - [Statistics]
File Window Execute Memory Configuration Statis
Total:
295 Cycle(s) executed.
ID executed by 132 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
fdvEX-Stages: 1, required Cycles: 19
Forwarding disabled.

Stalls:
RAW stalls: 141 (47.80% of all Cycles)
WAW stalls: 0 (0.00% of all Cycles)
Structural stalls: 0 (0.00% of all Cycles)
Control stalls: 19 (6.44% of all Cycles)
Trap stalls: 2 (0.68% of all Cycles)
Total: 162 Stall(s) (54.92% of all Cycles)

Conditional Branches):
Total: 13 (9.85% of all Instructions), thereof:
  taken: 7 (53.85% of all cond. Branches)
  not taken: 6 (46.15% of all cond. Branches)

Load-/Store-Instructions:
Total: 37 (28.03% of all Instructions), thereof:
  Loads: 25 (67.57% of Load-/Store-Instructions)
  Stores: 12 (32.43% of Load-/Store-Instructions)

Floating point stage instructions:
Total: 12 (9.09% of all Instructions), thereof:
  Additions: 0 (0.00% of Floating point stage inst.)
  Multiplications: 12 (100.00% of Floating point stage inst.)
  Divisions: 0 (0.00% of Floating point stage inst.)

Traps:
Traps: 1 (0.76% of all Instructions)
```

Question 1b

Number of clock cycle with forwarding : 186

Number of clock cycle without forwarding : 278

Speed up : $278 / 186 = 1.49$

With forwarding :

```
X - [Statistics]
File Window Execute Memory Configuration Statistics
Total:
186 Cycle(s) executed.
ID executed by 117 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: 48 (25.81% of all Cycles), thereof:
  LD stalls: 12 (25.00% of RAW stalls)
  Branch/Jump stalls: 6 (12.50% of RAW stalls)
  Floating point stalls: 30 (62.50% of RAW stalls)
WAW stalls: 0 (0.00% of all Cycles)
Structural stalls: 0 (0.00% of all Cycles)
Control stalls: 17 (9.14% of all Cycles)
Trap stalls: 3 (1.61% of all Cycles)
Total: 68 Stall(s) (36.56% of all Cycles)

Conditional Branches):
Total: 6 (5.13% of all Instructions), thereof:
  taken: 5 (83.33% of all cond. Branches)
  not taken: 1 (16.67% of all cond. Branches)

Load-/Store-Instructions:
Total: 37 (31.62% of all Instructions), thereof:
  Loads: 25 (67.57% of Load-/Store-Instructions)
  Stores: 12 (32.43% of Load-/Store-Instructions)

Floating point stage instructions:
Total: 12 (10.26% of all Instructions), thereof:
  Additions: 0 (0.00% of Floating point stage inst.)
  Multiplications: 12 (100.00% of Floating point stage inst.)
  Divisions: 0 (0.00% of Floating point stage inst.)

Traps:
Traps: 1 (0.85% of all Instructions)
```

Without forwarding:

```
X - [Statistics]
File Window Execute Memory Configuration Statistics
Total:
278 Cycle(s) executed.
ID executed by 117 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: 140 (50.36% of all Cycles)
WAW stalls: 0 (0.00% of all Cycles)
Structural stalls: 0 (0.00% of all Cycles)
Control stalls: 17 (6.12% of all Cycles)
Trap stalls: 3 (1.08% of all Cycles)
Total: 160 Stall(s) (57.55% of all Cycles)

Conditional Branches):
Total: 6 (5.13% of all Instructions), thereof:
  taken: 5 (83.33% of all cond. Branches)
  not taken: 1 (16.67% of all cond. Branches)

Load-/Store-Instructions:
Total: 37 (31.62% of all Instructions), thereof:
  Loads: 25 (67.57% of Load-/Store-Instructions)
  Stores: 12 (32.43% of Load-/Store-Instructions)

Floating point stage instructions:
Total: 12 (10.26% of all Instructions), thereof:
  Additions: 0 (0.00% of Floating point stage inst.)
  Multiplications: 12 (100.00% of Floating point stage inst.)
  Divisions: 0 (0.00% of Floating point stage inst.)

Traps:
Traps: 1 (0.85% of all Instructions)
```

Question 2a

Number of clock cycle with forwarding : 174

Number of clock cycle with forwarding in 1(b) : 186

Speed up : $186 / 174 = 1.068$

```
- [Statistics]
File Window Execute Memory Configuration Statistics
Total:
174 Cycle(s) executed.
105 Instruction(s) executed.
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: 60 (34.48% of all Cycles), thereof:
  LD stalls: 12 (20.00% of RAW stalls)
  Branch/Jump stalls: 6 (10.00% of RAW stalls)
  Floating point stalls: 42 (70.00% of RAW stalls)
WAW stalls: 0 (0.00% of all Cycles)
Structural stalls: 0 (0.00% of all Cycles)
Control stalls: 5 (2.87% of all Cycles)
Trap stalls: 3 (1.72% of all Cycles)
Total: 68 Stall(s) (39.08% of all Cycles)

Conditional Branches):
Total: 6 (5.71% of all Instructions), thereof:
  taken: 5 (83.33% of all cond. Branches)
  not taken: 1 (16.67% of all cond. Branches)

Load-/Store-Instructions:
Total: 37 (35.24% of all Instructions), thereof:
  Loads: 25 (67.57% of Load-/Store-Instructions)
  Stores: 12 (32.43% of Load-/Store-Instructions)

Floating point stage instructions:
Total: 12 (11.43% of all Instructions), thereof:
  Additions: 0 (0.00% of Floating point stage inst.)
  Multiplications: 12 (100.00% of Floating point stage inst.)
  Divisions: 0 (0.00% of Floating point stage inst.)

Traps:
Traps: 1 (0.95% of all Instructions)
```

Question 3a

Justification in rearranging code :

In previous code, instructions using same registers were consecutive. Thus because of lw and multiplication; stalling was happening. In this changed code, instructions using same registers are kept as away as possible to reduce this effect.

Number of clock cycle in scheduled code : 144

Number of clock cycle in unscheduled code : 174

Speed up : $174 / 144 = 1.20$

Question 3b

There are stalls because multiplication which can not be masked. It is because multiplication consumes large number of cycles and it is not possible to keep instructions further away without effecting each other.



```
X - [Statistics]
File Window Execute Memory Configuration Stat
Total:
144 Cycle(s) executed.
ID executed by 105 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: 30 (20.83% of all Cycles), thereof:
  LD stalls: 0 (0.00% of RAW stalls)
  Branch/Jump stalls: 0 (0.00% of RAW stalls)
  Floating point stalls: 30 (100.00% of RAW stalls)
WAW stalls: 0 (0.00% of all Cycles)
Structural stalls: 0 (0.00% of all Cycles)
Control stalls: 5 (3.47% of all Cycles)
Trap stalls: 3 (2.08% of all Cycles)
Total: 38 Stall(s) (26.40% of all Cycles)

Conditional Branches):
Total: 6 (5.71% of all Instructions), thereof:
  taken: 5 (83.33% of all cond. Branches)
  not taken: 1 (16.67% of all cond. Branches)

Load-/Store-Instructions:
Total: 37 (35.24% of all Instructions), thereof:
  Loads: 25 (67.57% of Load-/Store-Instructions)
  Stores: 12 (32.43% of Load-/Store-Instructions)

Floating point stage instructions:
Total: 12 (11.43% of all Instructions), thereof:
  Additions: 0 (0.00% of Floating point stage inst.)
  Multiplications: 12 (100.00% of Floating point stage inst.)
  Divisions: 0 (0.00% of Floating point stage inst.)

Traps:
Traps: 1 (0.95% of all Instructions)
```

Question 4

Explanation :

Without unrolling, during execution of checking termination for loop, execution instruction after this is started and must be aborted for every iteration of loop other than last one. Thus there is waste of this clock cycle which is utilized in unrolling.

Note : if we assume N is even, we do not need to check termination condition in middle of loop, thus number of clock cycles reduces from 171 to 162; and speed up increases from 1.017 to 1.074 (what to change is mentioned in code)

Number of clock cycle : 171 ($N/2 = 3$ clock cycles decrease, one for each iteration saved)

Number of clock cycle without unrolling code : 174 (in question 2)

Speed up : $174 / 171 = 1.017$



```
LX - [Statistics]
File Window Execute Memory Configuration Statistics
Total:
171 Cycle(s) executed.
ID executed by 105 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: 60 (35.10% of all Cycles), thereof:
  LD stalls: 12 (20.00% of RAW stalls)
  Branch/Jump stalls: 6 (10.00% of RAW stalls)
  Floating point stalls: 42 (70.00% of RAW stalls)
WAW stalls: 0 (0.00% of all Cycles)
Structural stalls: 0 (0.00% of all Cycles)
Control stalls: 2 (1.17% of all Cycles)
Trap stalls: 3 (1.75% of all Cycles)
Total: 65 Stall(s) (38.01% of all Cycles)

Conditional Branches):
Total: 6 (5.71% of all Instructions), thereof:
  taken: 2 (33.33% of all cond. Branches)
  not taken: 4 (66.67% of all cond. Branches)

Load-/Store-Instructions:
Total: 37 (35.24% of all Instructions), thereof:
  Loads: 25 (67.57% of Load-/Store-Instructions)
  Stores: 12 (32.43% of Load-/Store-Instructions)

Floating point stage instructions:
Total: 12 (11.43% of all Instructions), thereof:
  Additions: 0 (0.00% of Floating point stage inst.)
  Multiplications: 12 (100.00% of Floating point stage inst.)
  Divisions: 0 (0.00% of Floating point stage inst.)

Traps:
Trap: 1 (0.95% of all Instructions)
```

Question 5a

Assuming N is even.

Number of clock cycle : 90

Number of clock cycle without unrolling code : 174 (in question 2)

Speed up : $174 / 90 = 1.93$

In previous part since we were using same registers in unrolling, we could not keep instructions using same register further away then in part 3. However here we can use other registers and thus we can keep such instructions as away as possible. Also two mult must be as far as possible. Naming convention used is each Ri is renamed as R1i eg. R4 to R14 except counters.

Question 5b

Even after renaming, multiplication must not be overlapping. For that stall occurs which can not be masked away.

