Part A

Here the code fragment:

```
float sum=0;
float f[16];
int i=0;
do{
  sum += f[i] * f[i+8];
  i++;
}while(i!=8);
```

I assume that the compiler stores the variables starting with the address `0x0`.

Part B

The data file has the following columns: index, integer representation of float, float value:

```
Input of data memory:

00  0          == nan
01  1065353216 == 1.0
02  1073741824 == 2.0
03  1077936128 == 3.0
04  1082130432 == 4.0
05  1084227584 == 5.0
06  1086324736 == 6.0
07  1088421888 == 7.0
08  1090519040 == 8.0
09  1091567616 == 9.0
10  1092616192 == 10.0
11  1093664768 == 11.0
12  1094713344 == 12.0
13  1095761920 == 13.0
14  1096810496 == 14.0
15  1097859072 == 15.0
16  1098907648 == 16.0

Output of data memory:

00  1140195328 == 492.0
01  1065353216 == 1.0
02        ...
```

I used comments with the following syntax:

`-- Hazard-type: [Sub-type:] "Clck": From - To "Param:" Parameter-no Register Description.`

After the comment line follows the instruction line which was stalled in the Instruction Decoder because of the current hazard. The clock values are valid for the first iteration only.

```
ADDI R1 R0 0
loop:
```

```
3                      -- Data Hz: RAW: Clck: 5 - 8 Param: 1 R1 waiting for line 0
4  LF F1 4(R1)
5  LF F2 36(R1)
6                      -- Data Hz: RAW: Clck: 10 - 18 Param: 1 F1 waiting for line 4
7                      -- Data Hz: RAW: Clck: 10 - 19 Param: 2 F2 waiting for line 5
8  MULTF F3 F1 F2
9                      -- Data Hz: RAW: Clck: 20 - 25 Param: 2 F3 waiting for line 8
10 ADDF F0 F0 F3
11                     -- Struc. Hz: Clck: 26 - 27 ADDI (line 13) would be at the same
12                     --       time at MA as ADDF (line 10) therefore wait 1 cycle
13 ADDI R1 R1 4
14                     -- Data Hz: RAW: Clck: 28 - 31 Param: 1 R1 waiting for line 13
15 SEQI R3 R1 32
16                     -- Data Hz: RAW: Clck: 32 - 35 Param: 1 R3 waiting for line 15
17 BEQZ R3 loop
18 NOP
19 SF 0(R0) F0
20 TRAP 0
21 NOP
```

The complete execution took **295** clocks cycles.

Part C

I used the same comment style as above:

```
1  ADDI R1 R0 0
2  loop:
3                      -- Data Hz: RAW: Clck: 5 - 8 Param: 1 R1 waiting for line 0
4  LF F1 4(R1)
5  LF F2 36(R1)
6  ADDI R1 R1 4
7                      -- Data Hz: RAW: Clck: 15 - 18 Param: 1 F1 waiting for line 4
8                      -- Data Hz: RAW: Clck: 15 - 19 Param: 2 F2 waiting for line 5
9  MULTF F3 F1 F2
10 SEQI R3 R1 32
11                     -- Data Hz: RAW: Clck: 21 - 25 Param: 2 F3 waiting for line 9
12 ADDF F0 F0 F3
13                     -- Struc. Hz: Clck: 26 - 27 ADDF (line 12) would be at the same
14                     --       time at MA as BEQZ (line 17)
15                     -- Control. Hz: Clck: 26 - 27 ADDF (line 12) have to be finished
16                     --       before branch enter MA (BEQZ (line 17))
17 BEQZ R3 loop
18 NOP
19                     -- Data Hz: RAW: Clck: 29 - 30 Param: 2 F0 waiting for line 12
20 SF 0(R0) F0
21 TRAP 0
22 NOP
```

The last hazard in line 19 occurs each iteration but doesn't infect the runtime, because `SF 0(R0) F0` comes after the jump. Only after in the last iteration effects this hazard the runtime.

The execution time of this program is **219** cycles and the result in `F0` is as excepted 1140195328.

Part D

I used the same comment style as above:

```
1  ADDI R1 R0 0
2  loop:
3                  -- Data Hz: RAW: Clck: 5 - 8 Param: 1 R1 waiting for line 0
4  LF F1 4(R1)
5  LF F2 36(R1)
6  LF F3 8(R1)
7  LF F4 40(R1)
8                  -- Data Hz: RAW: Clck: 18 - 22 Param: 2 F2 waiting for line 5
9  MULTF F5 F1 F2
10 ADDI R1 R1 8
11 MULTF F6 F3 F4
12                 -- Data Hz: RAW: Clck: 28 - 31 Param: 2 F5 waiting for line 9
13 ADDF F0 F0 F5
14 SEQI R3 R1 32
15                  -- Data Hz: WAW: Clck: 34 - 36 Param: 0 F0 waiting for line 13
16                 -- Data Hz: RAW: Clck: 34 - 36 Param: 1 F0 waiting for line 13
17 ADDF F0 F0 F6
18                 -- Struc. Hz: Clck: 37 - 38 ADDF (line 17) would be at the same
19                 --        time at MA as BEQZ (line 22)
20                 -- Control. Hz: Clck: 37 - 38 ADDF (line 17) have to be finished
21                 --        before branch enter MA (BEQZ (line 22))
22 BEQZ R3 loop
23 NOP
24                 -- Data Hz: RAW: Clck: 40 - 41 Param: 2 F0 waiting for line 17
25 SF 0(R0) F0
26 TRAP 0
27 NOP
```

The execution time of this program is 159 cycles and the result in `F0` is as excepted 1140195328.

Comparison between the three programs: To compare the programs I want to take the time and the amount of registers into account. Latter is of interest because if the code is a part of a bigger program then you need additional cycles to store and restore registers if necessary. In the following table show some useful numbers:

	Part B	Part C	Part D
Time	295	219	159
Time %	100%	72%	54%
Register number	7	7	10
Register number %	100%	100%	143%
Time $\times$ Register no.	2065	1533	1590
Time $\times$ Register no. %	100%	74%	77%

The lower the values the better is the program. In conclusion is the Part C program a good trade of between memory and speed. If the machine is poor of registers than the Part C program is the best one and if the number of registers is not relevant than is Part D program is much better. The more knowledge about the concrete system is available the more powerful can be the optimization.

Part E

The code with counted and labeled instructions:

```
1  ADDI R1 R0 0
   loop:
2  LF F1 4(R1)
3  LF F2 36(R1)
4  ADDI R1 R1 4
5  MULTF F3 F1 F2
6  SEQI R3 R1 32
7  ADDF F0 F0 F3
8  BEQZ R3 loop
9  NOP
10 SF 0(R0) F0
11 TRAP 0
13 NOP
```

The following table shows for each clock the content of the Pipeline. The commands are counted from 1 to 11 as displayed in the code above.

Clock / Pipeline	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
IF	-	-	1a	2a	3a	3s	3s	3s	4a	5a	5s	5s	5s	6s	6s	6s	6s	6s	6s
ID	-	-	-	1a	2s	2s	2s	2a	3a	4a	4s	4s	4s	5s	5s	5s	5s	5s	5a
EX1	-	-	-	-	1a	-	-	-	2a	3a	3s	3s	3s	4s	4s	4s	4s	-	-
EX2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EX3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EX4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MA	-	-	-	-	-	1a	-	-	-	2a	2s	2s	2a	3a	3s	3s	3a	4a	-
WB	-	-	-	-	-	-	1a	-	-	-	-	-	-	2a	-	-	-	3a	4a

Clock / Pipeline	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
IF	7a	8a	8s	8s	8s	8s	9a	9s	10a	11a	-	-	-	2a	3a	4a	5a	5s	5s
ID	6a	7s	7s	7s	7s	7a	8s	8a	9a	10s	-	-	-	-	2a	3a	4a	4s	4s
EX1	5a	6a	-	-	-	-	7a	-	8a	9a	-	-	-	-	-	2a	3a	3s	3s
EX2	-	5a	-	-	-	-	-	7a	-	-	-	-	-	-	-	-	-	-	-
EX3	-	-	5a	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EX4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MA	-	-	6a	5a	-	-	-	-	7a	8a	9a	-	-	-	-	-	2a	2s	2s
WB	-	-	-	6a	5a	-	-	-	-	7a	-	-	-	-	-	-	-	-	-

Clock / Pipeline	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57
IF	5s	6s	6s	6s	6s	6s	6s	7a	8a	8s	8s	8s	8s	9a	9s	10a	11a	-	-
ID	4s	5s	5s	5s	5s	5s	5a	6a	7s	7s	7s	7s	7a	8s	8a	9a	10s	-	-
EX1	3s	4s	4s	4s	4s	-	-	5a	6a	-	-	-	-	7a	-	8a	9a	-	-
EX2	-	-	-	-	-	-	-	-	5a	-	-	-	-	-	7a	-	-	-	-
EX3	-	-	-	-	-	-	-	-	-	5a	-	-	-	-	-	-	-	-	-
EX4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MA	2a	3a	3s	3s	3a	4a	-	-	-	6a	5a	-	-	-	-	7a	8a	9a	-
WB	-	2a	-	-	-	3a	4a	-	-	-	6a	5a	-	-	-	-	7a	-	-