PIPELINED PROCESSOR

1. Assume that we have a pipelined processor with 5 stages and the following features:

- The Register File can be read and written in the same cycle.
- No forwarding technique is included.
- Branches are resolved at Decode. The next sequential instruction is fetched by default, and it is flushed if the branch is taken.

An application with the following features is executed:

- 20% of the instructions correspond to conditional branches, being 40% of them taken branches. There are not data dependences in branch instructions.
- 18% of the times, instruction I_{i+1} has a RAW dependency with instruction I_i (30% are load instructions).
- 6% of the times, instruction I_{i+2} has a RAW dependency with instruction I_i (30 % are load instructions), and in those cases there is never a dependency between I_{i+1} and I_{i+2}.

Obtain the following metrics:

a) CPI.
b) CPI if we add forwarding.
c) SPEED UP of b vs. a.

2. The following code is executed in a pipelined processor with 5 stages:

```
sub  $1,$2,$3
add  $4,$5,$6
sub  $5,$4,$8
add  $7,$2,$3
add  $9,$7,$3
lw   $1,10($6)
add  $3,$1,$4
sub  $6,$7,$8
```

Assuming that the branch register file can be read and written in the same cycle, calculate the necessary number of cycles for executing the code in the following cases:

a) There is neither forwarding nor instruction scheduling.
b) There is forwarding but no instruction scheduling.
c) There is no forwarding but there can be applied instruction scheduling.

3. Given a pipelined processor with 5 stages and the following features:

- It has the forwarding technique.
- It implements delayed branches with a "delay slot" equal to 1 cycle.
- It has a unified memory for instructions and data. Thus, it´s not possible to read an instruction and read/write data during the same cycle (MEM of a lw or sw cannot collide with IF of any instruction).
- The Register File can be read and written in the same cycle.

The following loop is executed in this processor:

```
LOOP:   LW R2,X(R6)
        LW R3,Y(R6)
        SUB   R2,R2,R3
        ADD   R2,R2,R1
        SUB   R6,R6,#4
        SW Z(R6),R2
        BNEZ  R6, LOOP
        ADD   R1,R1,#1
        SUB   R3,R3,R7
```

R6 initial value is 2000.

a) Represent the instruction/time diagram for the first iteration and highlight the forwardings that take place.
b) Calculate the CPI.
c) For a frequency of 1GHz, calculate the MIPS.

4. Given a pipelined processor with 5 stages and the following features:

- The Register File can be read and written in the same cycle.
- It has HW for detecting hazards and stall in Decode stage if necessary.
- It has the forwarding technique.
- Branches are resolved at Decode using stalls.
- Structural hazards at the MEM stage are detected and resolved via stalling in the last stage of execution.
- WAW hazards are resolved by avoiding the write of the first instruction.
- Functional units in the processor:

FU	Number	Latency	Pipelined
FP ADD	1	2	Yes
FP MUL	1	5	Yes
Int ALU	1	1	No

The following program is executed in this processor:

```
LD  F10,0(R1)
MULD  F4,F0,F10
LD  F12,0(R2)
ADDD F2,F12,F4
LD  F4 8(R1)
MULD  F12,F4,F12
LD  F12,16(R1)
```

a) Represent the instruction/time diagram for the program and highlight the forwardings that take place.
b) Calculate the CPI.

5. The following loop is executed in a pipelined processor with 7 stages:

```
LOOP: LD  F2, 0(R1)
      MULTD     F4, F2, F0
      LD  F6, 0(R2)
      ADDD   F6, F4, F6
      SD    0(R2), F6
      ADDI   R1, R1, 8
      SGTI    R3, R1, DONE   #R3=1 SI R1 ≥ DONE
      BEQZ   R3, LOOP
      ADDI   R2, R2, 8
```

Assuming that:

- Both data and instruction memory are pipelined in 2 stages.
- The Register File can be read and written in the same cycle.
- It has the forwarding technique.
- It has HW for detecting hazards and stall in Decode stage if necessary.
- Branches are resolved at Decode using stalls.
- Structural hazards at the MEM stage are detected and resolved via stalling in the last stage of execution.
- WAW hazards are resolved by avoiding the write of the first instruction.
- It implements delayed branches with a "delay slot" equal to 1 cycle.
- Arithmetic and store instructions can coexist at MEM and WB.
- Functional units in the processor:

UF	Number	Latency	Pipelined
FP ADD	1	4	Yes
FP MUL	1	5	Yes
Int ALU	1	1	No

a) Determine the CPI, considering that done=0x1000 and r1=0x0100 at the beginning.
b) Schedule the code for minimizing the CPI.

6. Given a DLX microprocessor with the following characteristics:

- Pipelined processor with 5 stages: IF, ID, EX, MEM, WB.
- The Register File can be read and written in the same cycle.
- It has HW for detecting hazards and for resolving them via bypass. RAW dependencies are detected at the ID Stage and are resolved via bypass as soon as the operands are available.
- Branches are resolved at Decode using stalls. For branch instructions, bypass to the ID Stage is also supported. It implements delayed-branches with a delay-slot equal to 1 cycle.
- All bypasses are carried out from the pipeline registers.
- Structural hazards at the MEM stage are detected and resolved via stalling in the last stage of execution.
- WAW hazards are resolved by avoiding the write of the first instruction.
- Functional units in the processor:

Functional Unit	#	Latency	Pipelined
FP ADD/SUB	1	2	Yes
FP MUL	1	3	Yes
Integer ALU	1	1	No

The following program is executed in that processor. The program has 2 arrays, initialized as: A={5'5, 7'2, -2'8, 8'1} and B={-207'8, 9'3, -3'5, 13'4}, and stores the result in a third vector C. This program has been generated using a compiler with no optimizations, so the delay-slot is naively filled with a NOP instruction in all cases.

```
LOOP:   LD      F1,0(R3)
        LD      F2,0(R4)
        MULD        F3,F1,F2
        ADDI        R4,R4,#8
        ANDI        R6,R1,#1
        BEQ     R6,R0,NO_ADD
        NOP
        ADDD        F3,F3,F4
NO_ADD:
        ADDI        R1,R1,#1
        SUBD        F3,F3,#4
        SD      F3,0(R5)
        ADDI        R3,R3,#8
        ADDI        R5,R5,#8
        BNE     R1,R2,LOOP
        NOP
```

Assume the following initial values for the registers:

- R0=0, R1=0, R2=4
- R3= Initial address of vector A
- R4= Initial address of vector B
- R5= Initial address of vector C
- F4=1

a) Represent the instruction/time diagram for the program and highlight the bypasses that take place. Explain explicitly the stalls and their causes.
b) Calculate the CPI.
c) Write the program provided by a compiler which fills the delay-slot in a more efficient

way. What would be the impact of this optimization on the CPI?

d) Reorder the program for minimizing the CPI, including the optimizations performed in the previous section. You don't need to represent the new instruction/time diagram nor calculate the new CPI.

7. In the data path in the following figure (pipelined MIPS as explained in the basic pipelining slides) all control signals are named using the stage as suffix, i.e. WriteRegM belongs to the Memory stage. Forwarding and Hazard detection units are performed in the module labelled as Hazard Unit.

a) Obtain the expression for the control signals of multiplexers in the inputs of the ALU: ForwardAE y ForwardBE. You can use == and != to compare multibit values.

b) Design the logic to compute ForwardAE using combinationa blocks and logic gates.

c) Obtain the expression of the signals StallF, StalD y FlushE, used to stall the stages IF and ID

d) Obtain the expression of ForwardBD y ForwardAD.

e) Is there any problemais the following instructions are executed?

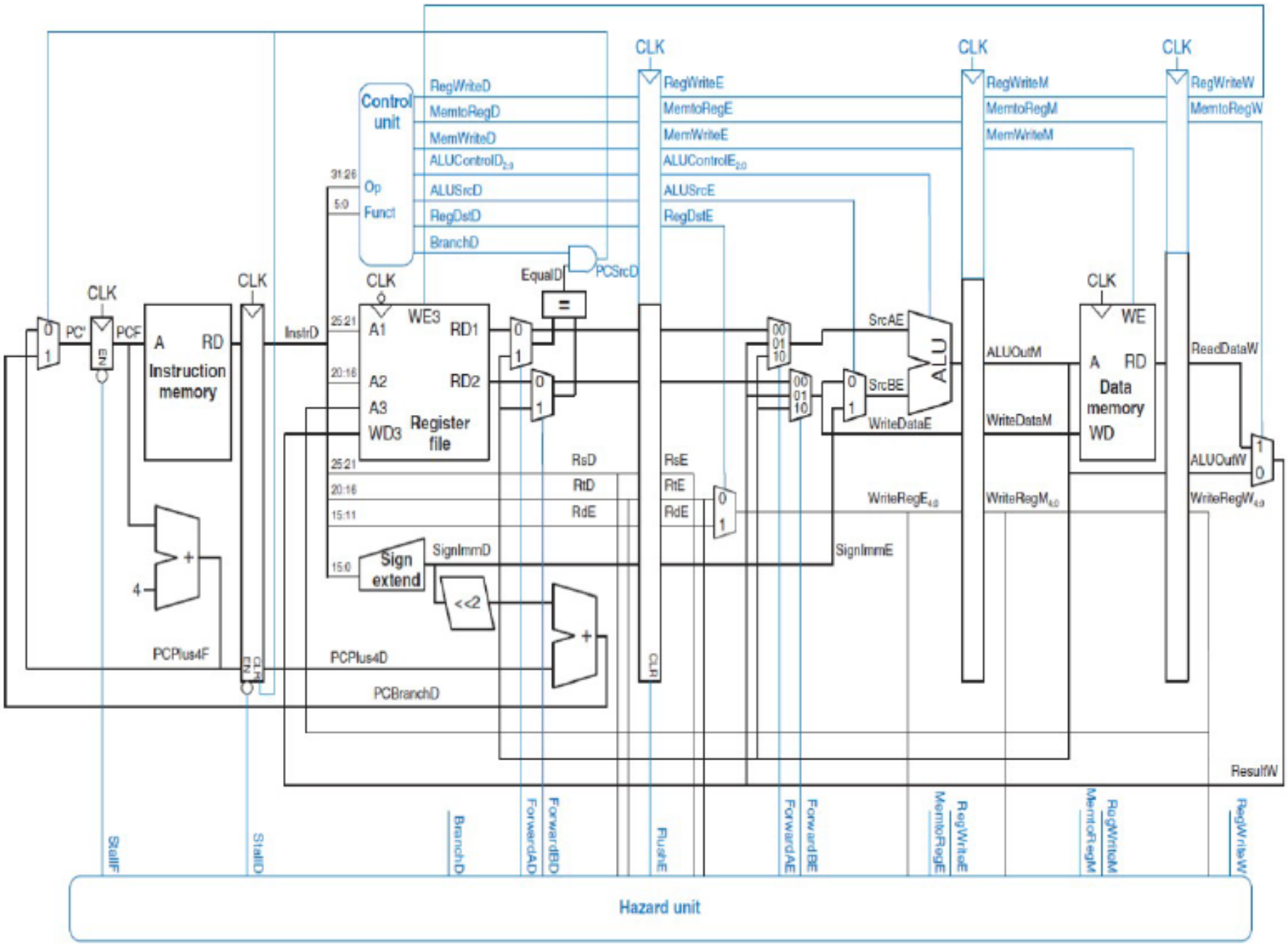
```
lw r1, 0(r3)
beq r1, r0, 40
```

How can the problem be solved?

f) Assume the following code is executed:

```
LOOP: LW R2,X(R6)
      LW R3,Y(R6)
      SUB R2,R2,R3
      ADD R2,R2,R1
      SUB R6,R6,#4
      SW Z(R6),R2
      BNEZ R6,LOOP
      ADD R1,R1,#1
```

What will be the value of the signals in cycle 5 (when the first instruction reaches WB state)?


CLK
Control unit
Op
Funct
RegWriteD
MemtoRegD
MemWriteD
ALUControlD2:0
ALUSrcD
RegDstD
BranchD
EqualD
PCSrcD
RegWriteE
MemtoRegE
MemWriteE
ALUControlE2:0
ALUSrcE
RegDstE
RegWriteM
MemtoRegM
MemWriteM
RegWriteW
MemtoRegW
PC'
PCF
A
RD
Instruction memory
InstrD
31:26
5:0
25:21
20:16
15:11
15:0
WE3
A1
A2
A3
WD3
RD1
RD2
Register file
RsD
RtD
RdE
RsE
RtE
RdE
Sign extend
SignImmD
SignImmE
<<2
4
PCPlus4F
PCPlus4D
PCBranchD
SrcAE
SrcBE
WriteDataE
ALU
ALUOutM
WriteDataM
WE
Data memory
WD
ReadDataW
ALUOutW
WriteRegE4:0
WriteRegM4:0
WriteRegW4:0
ResultW
StallF
StallD
BranchD
ForwardAD
ForwardBD
FlushE
ForwardAE
ForwardBE
RegWriteE
MemtoRegE
RegWriteM
MemtoRegM
RegWriteW
Hazard unit