EXERCISE 1 (F): SCOREBOARD

Given the following loop taken from a high level program:

```
do      {
         BASEC[i] = BASEA[i] + BASEB[i] + INC1 + INC2;
         i++;
        }
while (i != N)
```

The program has been compiled in MIPS assembly code assuming that registers $4 and $7 have been initialized with values 0 and 4N respectively. The symbols BASEA, BASEB and BASEC are 16-bit constant. The processor clock cycle is 0.5 ns.

```
L1:     lw   $2, BASEA ($4)
        addi $2, $2, INC1
        lw   $3, BASEB ($4)
        addi $3, $3, INC2
        add  $5, $2, $3
        sw   $5, BASEC ($4)
        addi $4, $4, 4
        bne  $4, $7, L1
```

We assume the original program be executed on a CPU with dynamic scheduling based on the SCOREBOARD technique with:

- 2 LOAD/STORE Functional Units (LDU1, LDU2) with latency 2 cycles
- 2 ALU/BR Functional Units (ALU1, ALU2) with latency 1 cycle
- Register File has 2 Read Ports and 1 Write Port
- Check structural hazards in ISSUE phase
- Check RAW hazards and RF READ in READ OPERANDS phase
- Check WAR e WAW and RF WRITE n WRITE BACK phase
- No forwarding
- Static branch prediction for backward branches: branch always taken

Please complete the scoreboard table by assuming all cache hits:

INSTRUCTION	ISSUE	READ OPERANDS	EXEC COMPLETE	WRITE BACK	Hazard Type	UNIT
`L1:lw $2,BASEA($4)`	1	2	4	5	*(CNTR hazard ok)*	LDU1
`addi $2, $2, INC1`	2	6	7	8	**RAW in $2** *(WAW $2 ok)*	ALU1
`lw $3,BASEB($4)`	3	4	6	7		LDU2
`addi $3,$3,INC2`	4	8	9	10	**RAW in $3** *(WAW $3 ok)*	ALU2
`add $5,$2,$3`	9	11	12	13	**Struct ALU/BR FU**+ **RAW $3**	ALU1
`sw $5,BASEC($4)`	10	14	16	17	**RAW $5**	LDU1
`addi $4,$4,4`	11	12	13	15	**WAR $4**	ALU2
`bne $4,$7, L1`	14	16	17	18	**Struct ALU/BR FU** + **RAW $4**	ALU1
`EXIT:`						

- Given a clock frequency of 2 GHz, calculate the following metrics:
  - Instruction Count per iteration (IC) = 8
  - CPI = # cycles / IC = 18 / 8 = 2.25
  - Throughput (expressed in MIPS): MIPS = f_{CLOCK} / (CPI * 10^6) = ($2 * 10^9$) / ($2.25 * 10^6$) = 888.88

EXERCISE 1 (G): SCOREBOARD

We assume the original program be executed on a CPU with dynamic scheduling based on the SCOREBOARD technique with:

- 2 LOAD/STORE Functional Units (LDU1, LDU2) with latency 2 cycles
- 3 ALU/BR Functional Units (ALU1, ALU2, ALU3) with latency 1 cycle
- Register File has 4 Read Ports and 2 Write Ports
- Check structural hazards in ISSUE phase
- Check RAW hazards and RF READ in READ OPERANDS phase
- Check WAR e WAW and RF WRITE n WRITE BACK phase
- Forwarding
- Static branch prediction for backward branches: branch always taken

Please complete the scoreboard table by assuming all cache hits:

ISTRUZIONE	ISSUE	READ OPERANDS	EXEC COMPLETE	WRITE BACK	Hazards Type	FORWARDING	UNIT
`L1:lw $2,BASEA($4)`	1	2	4	5	*(CNTR hazard ok)*		LDU1
`addi $2, $2, INC1`	2	5	6	7	**RAW $2** *(WAW $2 ok)*	**Forw $2**	ALU1
`lw $3,BASEB($4)`	3	4	6	7	*(RF write ok)*		LDU2
`addi $3,$3,INC2`	4	7	8	9	**RAW in $3** *(WAW $3 ok)*	**Forw $3**	ALU2
`add $5,$2,$3`	5	9	10	11	**RAW $2, RAW $3**	**Forw $3**	ALU3
`sw $5,BASEC($4)`	6	11	13	14	**RAW $5**	**Forw $5**	LDU1
`addi $4,$4,4`	8	9	10	12	**Struct ALU/BR FU** + **WAR $4**		ALU1
`bne $4,$7, L1`	10	11	12	13	**Struct ALU/BR FU** *(RAW $4 ok)*	**Forw $4**	ALU2
`EXIT:`							

- Calculate the speedup with respect to the previous Scoreboard (EX 1.F):

  Speedup = (Exec. Time Scoreboard 1.F) / (Exec. Time Scorebord New) = 18/14= 1,29

EXERCISE 2 (D): SCOREBOARD

Given the following loop expressed in a high level language:

```
for (i =0; i < N; i ++)
      vectC[i] = vectA[i] + vectB[i];
```

The program has been compiled in MIPS assembly code assuming that registers `$t6` and `$t7`have been initialized with values 0 and 4N respectively. The symbols VECTA, VECTB and VECTC are 16-bit constant. The processor clock frequency is **1 GHz.**

INSTRUCTION	Comment
FOR1:beq $t6,$t7, END	# if ($t6 == $t7) goto END
lw $t2,VECTA($t6)	# $t2 <- VECTA [$t6];
lw $t3,VECTB($t6)	# $t3 <- VECTB [$t6];
add $t2,$t2,$t3	# $t2 <- $t2 + $t3;
sw $t2,VECTC($t6)	# VECTC[$t6] <- $t2;
addi $t6,$t6,4	# $t6 <- $t6 + 4;
j FOR1	# goto FOR1;
END:	

Assuming the program be executed by a CPU with dynamic scheduling based on SCOREBOARD with:

- 2 LOAD/STORE units (LDU1, LDU2) with latency 4 cycles when cache hits
- 2 ALU/BR/J units (ALU1, ALU2) with latency 2 cycles
- Register File has 2 Read Ports and 1 Write Port
- Check structural hazards in ISSUE phase
- Check RAW hazards and RF READ in READ OPERANDS phase
- Check WAR e WAW and RF WRITE in WRITE BACK phase
- Forwarding
- Static Branch Prediction BTFNT (BACKWARD TAKEN FORWARD NOT TAKEN) with Branch Target Buffer

Please complete the SCOREBOARD TABLE by assuming all cache HITS:

ISTRUZIONE	PRED. T / NT	ISSUE	READ OPERANDS	EXECUTION COMPLETE	WRITE BACK	HAZARDS TYPE	Forwarding	UNIT
`FOR1:beq $t6,$t7, FOR2`	NT	1	2	4	5	*(CNTR hazard ok)*		ALU1
`lw $t2,VECTA($t6)`	-	2	3	7	8	*(CNTR hazard ok)*		LDU1
`lw $t3,VECTB($t6)`	-	3	4	8	9			LDU2
`add $t2,$t2,$t3`	-	4	9	11	12	**RAW $2, RAW $t3**	**Forw $t3**	ALU2
`sw $t2,VECTC($t6)`	-	9	12	16	17	**STRUCT LDU1, RAW $t2**	**Forw $t2**	LDU1
`addi $t6,$t6,4`	-	10	11	13	14	*(WAR $t6 OK)*		ALU1
`j FOR1`	T	13	14	16	17*	**STRUCT ALU2**		ALU2

`(*) JUMP does not write the RF in Write Back phase`

- Given a clock frequency of 2 GHz, calculate the following metrics:
  - Instruction Count (IC) = 7
  - CPI = # cycles / IC = 17 / 7 = 2.43
  - IPC = 1 / CPI = 0.41
  - Throughput (expressed in MIPS): MIPS = f_{CLOCK} / (CPI * 10^6) = (2 * 10^9) / (2.43 * 10^6) = 823

EXERCISE 2 (E): SCOREBOARD with DATA CACHE MISSES

Assuming the program be executed by a CPU with dynamic scheduling based on SCOREBOARD with:

- 2 LOAD/STORE units (LDU1, LDU2) with Latency 4 cycles
- 2 ALU/BR/J units (ALU1, ALU2) with Latency 2 cycles
- Register File has 2 Read Ports and 1 Write Port
- DATA MEMORY with latency 10 cycles when DATA CACHE MISS
- Check structural hazards in ISSUE phase
- Check RAW hazads in READ OPERANDS phase
- Check WAR e WAW in WRITE BACK phase
- Forwarding
- Static Branch Prediction BTFNT (BACKWARD TAKEN FORWARD NOT TAKEN) with Branch Target Buffer

Please complete the SCOREBOARD TABLE by assuming all data cache **MISSES**:

ISTRUZIONE	PRED. T / NT	ISSUE	READ OPERANDS	EXECUTION COMPLETE	MEM ACCESS COMPLETE	WRITE BACK	HAZARDS TYPE	Forwarding	UNIT
`FOR1:beq $t6,$t7, FOR2`	NT	1	2	4	-	5	*(CNTR hazard ok)*		ALU1
`lw $t2,VECTA($t6)`	-	2	3	7	17	18	*(CNTR hazard ok)*		LDU1
`lw $t3,VECTB($t6)`	-	3	4	8	18	19			LDU2
`add $t2,$t2,$t3`	-	4	19	21	-	22	**RAW $2, RAW $t3**	**Forw $t3**	ALU2
`sw $t2,VECTC($t6)`	-	19	22	26	36	37	**STRUCT LDU1, RAW $t2**	**Forw $t2**	LDU1
`addi $t6,$t6,4`	-	20	21	23	-	24	*(WAR $t6 OK)*		ALU1
`j FOR1`	T	23	24	26	-	27	**STRUCT ALU2**		ALU2

- Calculate the performance impact of the data cache misses with respect to the previous case (EX 4):

  (Exec. Time Scoreboard 4) / (Exec. Time Scorebord 5) = 17/37 = 0.46