Computer Architecture Recitation 2

BLG 322E

10.04.2025

Question 1

Consider the exemplary RISC processor given in Section 2.4.2 of the lecture notes. Differing from lecture notes, suppose that the instruction pipeline is designed with four stages as explained below:

1. **Instruction Fetch and Decode (ID):** Instruction fetch and instruction decode
2. **Read registers (RR):** Read registers (operands)
3. **Execute (EX):** Perform ALU operation, compute jump/branch targets, make branch decisions
4. **Memory, Write back (MW):** Two stages in the original processor have been combined; the ME/WB register has been removed

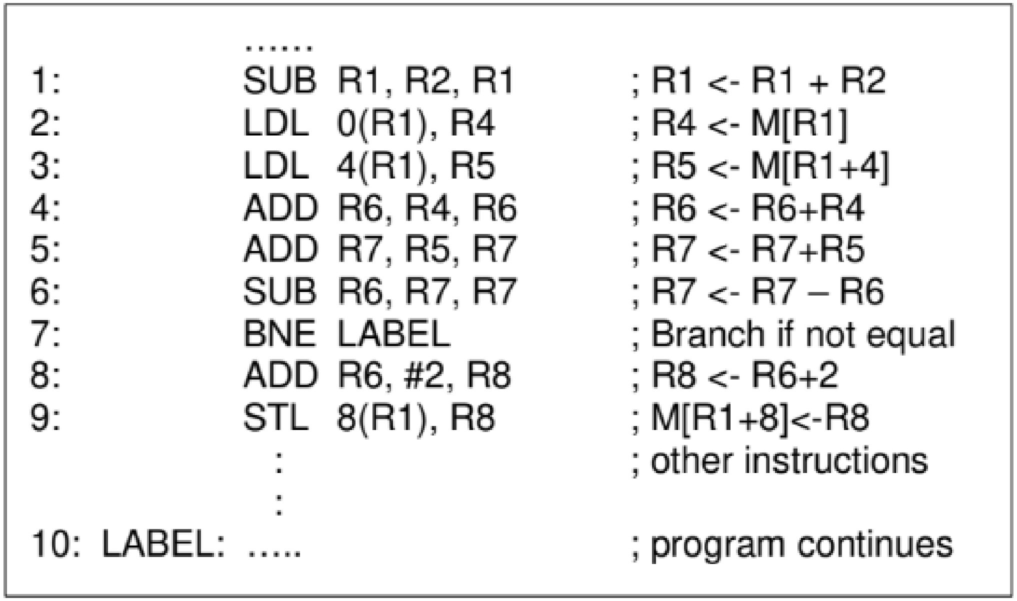
Question 1

Assume the following:

- The register file access hazard is NOT fixed.
- The processor does NOT have any forwarding (bypass) connections.
- The internal structure of the execution stage is similar to the circuitry shown on slide 2.53 (latest version, 2021), i.e., branch target address calculation and decision operations are performed in the EX stage, and results are sent directly to the ID stage.

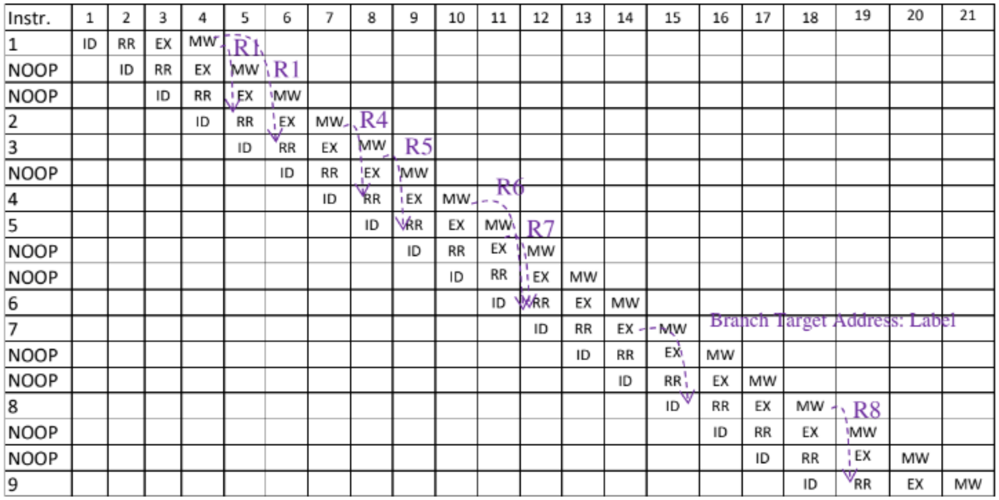
Question 1a

Draw the timing diagram for the piece of code given below to show the **data and branch hazards** and the **NOOP instructions that need to be inserted** to prevent these hazards using a **software-based solution**.



```
                 ......
1:               SUB  R1, R2, R1          ; R1 <- R1 + R2
2:               LDL  0(R1), R4           ; R4 <- M[R1]
3:               LDL  4(R1), R5           ; R5 <- M[R1+4]
4:               ADD  R6, R4, R6          ; R6 <- R6+R4
5:               ADD  R7, R5, R7          ; R7 <- R7+R5
6:               SUB  R6, R7, R7          ; R7 <- R7 – R6
7:               BNE  LABEL               ; Branch if not equal
8:               ADD  R6, #2, R8          ; R8 <- R6+2
9:               STL  8(R1), R8           ; M[R1+8]<-R8
                   :                      ; other instructions
                   :
10: LABEL:  .....                         ; program continues
```

Solution 1a


R1
R1
R4
R5
R6
R7
Branch Target Address: Label
R8

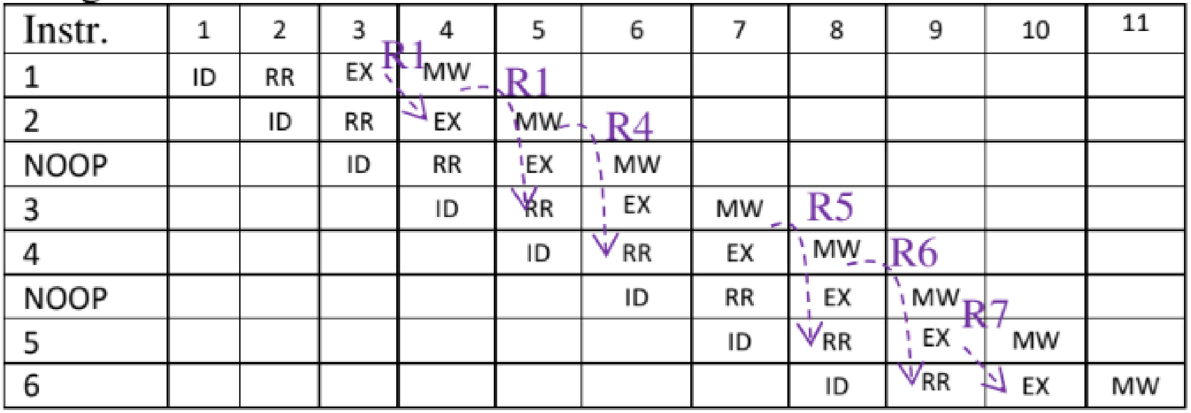
Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	ID	RR	EX	MW																	
NOOP		ID	RR	EX	MW																
NOOP			ID	RR	EX	MW															
2				ID	RR	EX	MW														
3					ID	RR	EX	MW													
NOOP						ID	RR	EX	MW												
4							ID	RR	EX	MW											
5								ID	RR	EX	MW										
NOOP									ID	RR	EX	MW									
NOOP										ID	RR	EX	MW								
6											ID	RR	EX	MW							
7												ID	RR	EX	MW						
NOOP													ID	RR	EX	MW					
NOOP														ID	RR	EX	MW				
8															ID	RR	EX	MW			
NOOP																ID	RR	EX	MW		
NOOP																	ID	RR	EX	MW	
9																		ID	RR	EX	MW

Question 1b

- Assume that there is operand forwarding (bypassing) from the output of the EX stage to the inputs of the ALU.
- Other parts of the CPU are not modified.
- Consider only the lines **between 1 and 6 (including 6)** of the piece of code given in Part (a).
- Draw the timing diagram for the piece of code **(lines 1-6)** to show **hazardss** and the **NOOP instructions that need to be inserted** to prevent these hazards using a **software-based solution**.

Solution 1b

Straight arrows from EX to EX show how the conflicts are resolved using bypassing.


R1
R1
R4
R5
R6
R7

Instr.	1	2	3	4	5	6	7	8	9	10	11
1	ID	RR	EX	MW							
2		ID	RR	EX	MW						
NOOP			ID	RR	EX	MW					
3				ID	RR	EX	MW				
4					ID	RR	EX	MW			
NOOP						ID	RR	EX	MW		
5							ID	RR	EX	MW	
6								ID	RR	EX	MW

Question 2

Consider a MIPS-like, 5 stage RISC processor:

- **IF:** Fetch instruction from instruction memory, calculate next PC.
- **ID:** Decode instruction and read source operands from register file.
- **EX:** Execute instruction depending on type:
  - arithmetic: perform arithmetic operation is performed, update flags
  - memory: calculate address
  - branch: evaluate condition and calculate starget address.
- **MEM:** Access memory (if instruction is a memory operation)
- **WR:** Write to register file
  - Write either result from ALU or data from memory to registers

Question 2

- This processor does not have any forwarding (bypass) connections.
- The register file hazard is fixed: write to register file in first half of clock cycle, and read in second half.
- Branch target address calculation and decision operations performed in EX, results sent directly to IF.
- Hardware stalls ares used for all pipeline hazards (conflicts).

```
BEGIN:  SUB  R2, R2, R2      ; R2 <- R2 – R2
        ADD  R2, #2, R3      ; R3 <- R2 + 2
        ADD  R1, #8, R1      ; R1 <- R1 + 8

LOOP:  LDL   0(R1), R4       ; R4 <- M[R1]
       ADD   R2, R4, R2      ; R2 <- R2 + R4
       SHL   R1, 2, R1       ; R1 <- R1 << 2    (Shift left by 2)
       ADD   R3, #-1, R3     ; R3 <- R3 + (-1)
       BNZ   LOOP            ; BRANCH IF NOT ZERO
       -----
```

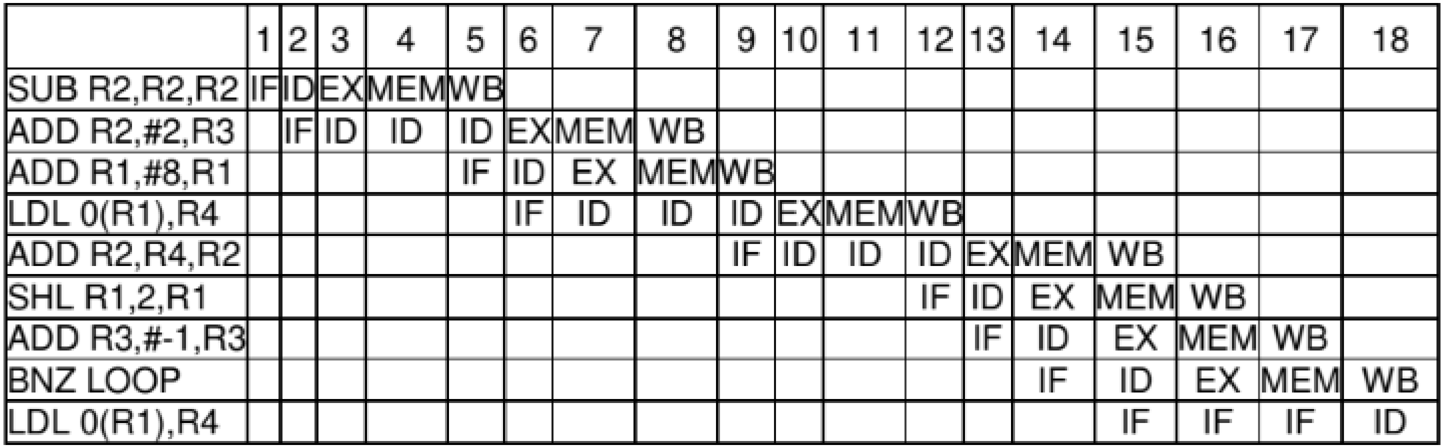
Question 2

a) Draw the space-time diagram from `BEGIN` until the end of the first iteration of the loop (including `BNZ` instruction). In the 9th cycle, which register(s) does the CPU read from and which register(s) does it write to?

b) We can calculate **Cycles Per Instruction (CPI)** as the total number of execution cycles divided by the total number of instructions. What is the CPI for this code when the execution of all iterations of the loop finishes?

c) To minimize penalty, find **optimized** solutions to all conflicts, if possible. (Do not change the algorithm, and make sure your solution does not affect the correctness of the program.) How many cycles does it take to complete the **first** iteration of the loop (starting from `BEGIN` and including `BNZ`) with the new solution?

Solution 2a



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
SUB R2,R2,R2	IF	ID	EX	MEM	WB													
ADD R2,#2,R3		IF	ID	ID	ID	EX	MEM	WB										
ADD R1,#8,R1					IF	ID	EX	MEM	WB									
LDL 0(R1),R4						IF	ID	ID	ID	EX	MEM	WB						
ADD R2,R4,R2									IF	ID	ID	ID	EX	MEM	WB			
SHL R1,2,R1												IF	ID	EX	MEM	WB		
ADD R3,#-1,R3													IF	ID	EX	MEM	WB	
BNZ LOOP														IF	ID	EX	MEM	WB
LDL 0(R1),R4															IF	IF	IF	ID

CPU writes to register R1 and reads from it in the 9th cycle.

Solution 2b

The loop iterates twice.

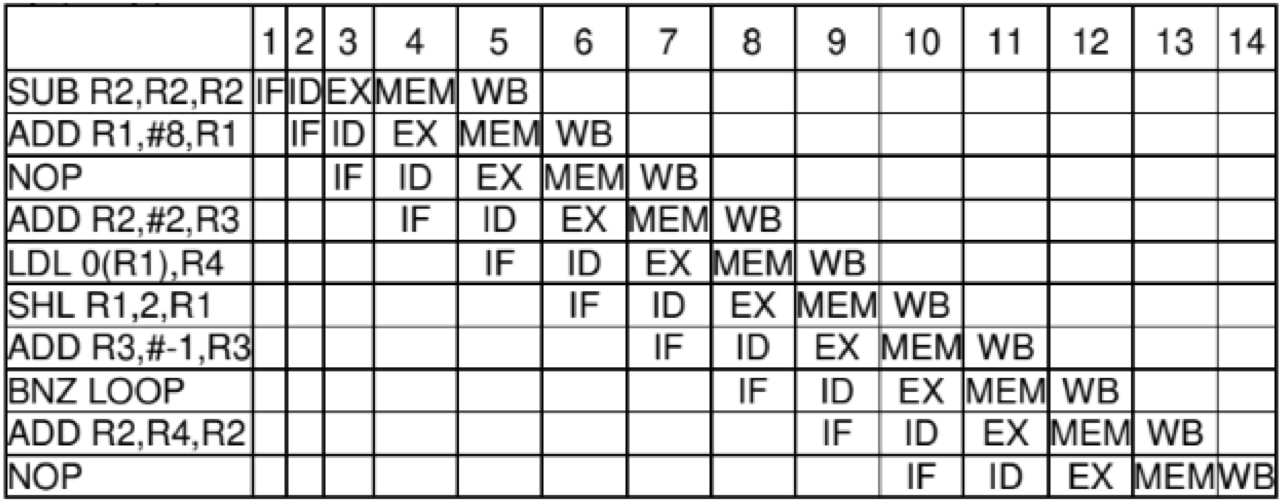
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
SUB R2,R2,R2	IF	ID	EX	MEM	WB																						
ADD R2,#2,R3		IF	ID	ID	ID	EX	MEM	WB																			
ADD R1,#8,R1					IF	ID	EX	MEM	WB																		
LDL 0(R1),R4						IF	ID	ID	ID	EX	MEM	WB															
ADD R2,R4,R2									IF	ID	ID	ID	EX	MEM	WB												
SHL R1,2,R1												IF	ID	EX	MEM	WB											
ADD R3,#-1,R3													IF	ID	EX	MEM	WB										
BNZ LOOP														IF	ID	EX	MEM	WB									
LDL 0(R1),R4															IF	IF	IF	ID	EX	MEM	WB						
ADD R2,R4,R2																		IF	ID	ID	ID	EX	MEM	WB			
SHL R1,2,R1																					IF	ID	EX	MEM	WB		
ADD R3,#-1,R3																						IF	ID	EX	MEM	WB	
BNZ LOOP																							IF	ID	EX	MEM	WB

$$\text{Total number of instructions executed} = 13$$

$$\text{Total number of cycles} = 27$$

$$\therefore CPI = 27/13$$

Solution 2c



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
SUB R2,R2,R2	IF	ID	EX	MEM	WB									
ADD R1,#8,R1		IF	ID	EX	MEM	WB								
NOP			IF	ID	EX	MEM	WB							
ADD R2,#2,R3				IF	ID	EX	MEM	WB						
LDL 0(R1),R4					IF	ID	EX	MEM	WB					
SHL R1,2,R1						IF	ID	EX	MEM	WB				
ADD R3,#-1,R3							IF	ID	EX	MEM	WB			
BNZ LOOP								IF	ID	EX	MEM	WB		
ADD R2,R4,R2									IF	ID	EX	MEM	WB	
NOP										IF	ID	EX	MEM	WB

Question 3a

- A CPU that has an instruction pipeline with branch prediction runs a program containing a conditional branch instruction such as "BZ - Branch if zero" that is executed many times.
- We check the decision of the prediction strategy in any six executions (runs) of the instruction and observe the following pattern:

#run	1	2	3	4	5	6
Prediction	N	N	T	N	T	T

N: Predict NOT to take the branch
T: Predict to take the branch

i) Which of the 2-bit dynamic prediction strategies covered in the lectures is used in this system? Draw the state transition diagram. Explain your answer.

ii) Consider how the predictions change, and fill in the table below that shows the prediction bits and whether the branch is really taken or not in each of the six runs. Fill in the last row only for the first five runs of the BZ instruction.

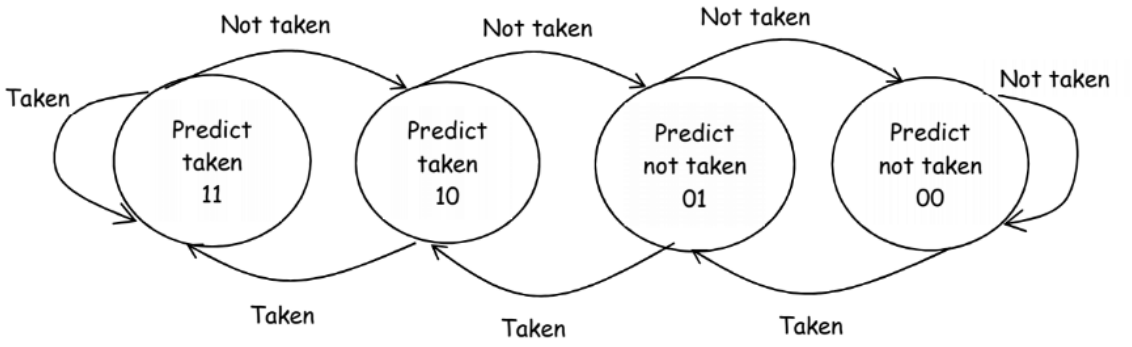
#run	1	2	3	4	5	6
Prediction	N	N	T	N	T	T
Prediction bits						
Really **T**aken or **N**ot						X

Solution 3a

i) A **saturating counter** is used. From Run 3 to Run 4, we cannot immediately transition from **Taken** to **Not Taken** if the branch is not actually taken and if we do not use a saturating counter.

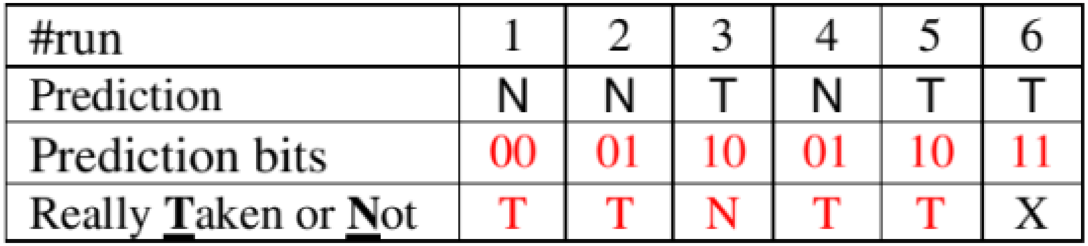
#run	1	2	3	4	5	6
Prediction	N	N	T	N	T	T

N: Predict NOT to take the branch
T: Predict to take the branch



Solution 3a

ii)

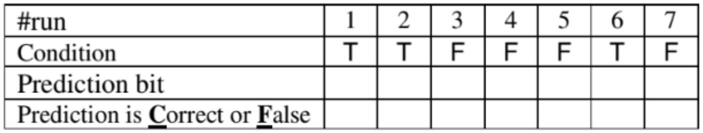


#run	1	2	3	4	5	6
Prediction	N	N	T	N	T	T
Prediction bits	00	01	10	01	10	11
Really **T**aken or **N**ot	T	T	N	T	T	X

Question 3b

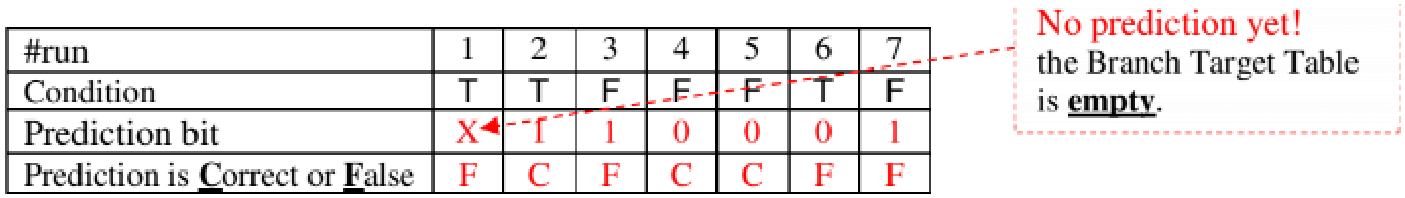
- A CPU that has an instruction pipeline with branch prediction runs a program containing a conditional branch instruction such as “BZ - Branch if zero” that is executed 7 times in the program.
- We check the condition (**“if zero”**) in each execution of the instruction and observe the following pattern:

  T T F F F T F , where **T**: True, **F**: False
- This system uses a 1-bit dynamic branch prediction strategy, and in the beginning, the Branch Target Table is empty.
- Consider how the condition changes, and fill in the table below that shows the prediction bit and whether the prediction is correct or not for each run of the BZ instruction.



#run	1	2	3	4	5	6	7
Condition	T	T	F	F	F	T	F
Prediction bit							
Prediction is <u>C</u>orrect or <u>F</u>alse							

Solution 3b



#run	1	2	3	4	5	6	7
Condition	T	T	F	F	F	T	F
Prediction bit	X	1	1	0	0	0	1
Prediction is **C**orrect or **F**alse	F	C	F	C	C	F	F

No prediction yet!
the Branch Target Table is **empty**.