Conditional Branch Instruction

What are branch instruction? Explain, how a branch instruction works with an example and suitable diagrams.

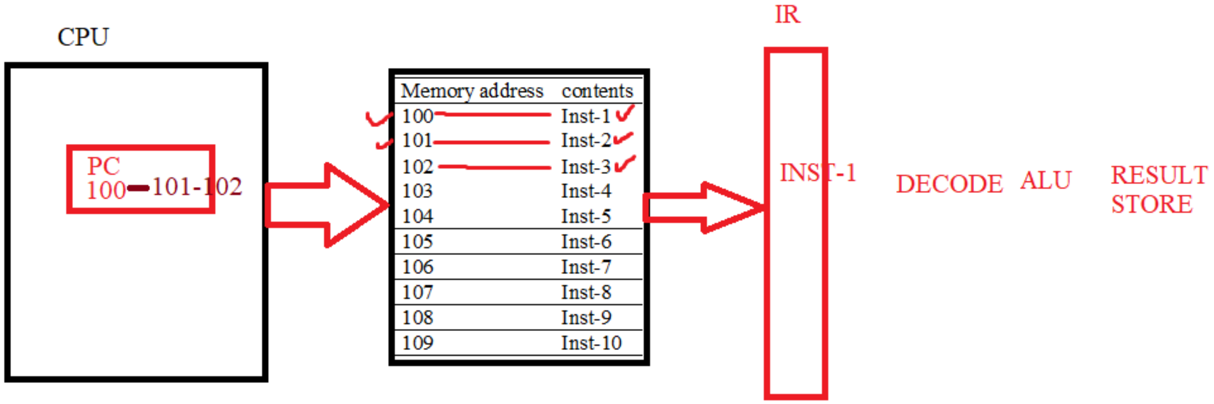
The branch instructions are used to change the sequence of instruction execution. To solve problems or to implement different types of algorithms, users/programmers use branch instructions to change the sequence of instruction execution in a program by skipping some instructions or calling functions etc..

A program contains 10 instructions, each of one Byte and loaded into a byte addressable RAM starting at memory address 100.

If there is no branch instruction in the program, the CPU will fetch & process instructions in sequence as it appears in the program and loaded in memory location. It means, the CPU will fetch & process Inst-1, followed by Inst-2, Inst-3, Inst-4 so on and continue until the end of the list.

Memory address	contents
100	Inst-1
101	Inst-2
102	Inst-3
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

each inst of 1 Byte
RAM byte addressable



Memory address	contents
100	Inst-1 ✓
101	Inst-2 ✓
102	Inst-3 ✓
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

Branch instructions are divided into two categories : (i) Unconditional Branch and (ii) Conditional branch instructions.

The general format of Unconditional Branch instructions is as follows

Opcode	Target Address (Memory address to read instruction)
JUMP	106

A program contains 10 instructions, each of one Byte and loaded into a byte addressable RAM starting at memory address 100.

Inst-3 is an unconditional branch instruction:

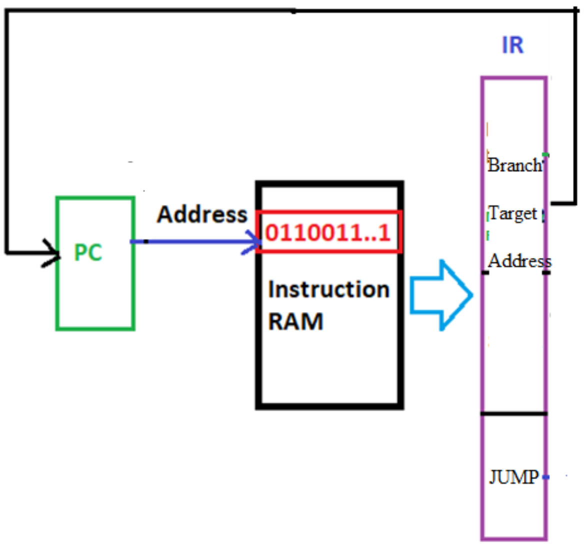
JUMP 106

When the program is run, CPU fetches & executes Inst-1 followed by Inst-2 and Inst-3. Here Inst-3 is an unconditional Jump instruction and it instructs the CPU to Jump to memory address 106 (Target Address) to read next instruction. So after processing Inst-3, the CPU changes the sequence of instructions, that means it does not process Inst-4, Inst-5 rather it Jumps to Inst-7. The CPU will read Inst-7 and process it. Moreover the CPU will continue to process following instruction in sequence from memory address 106 until there appears another conditional instruction.

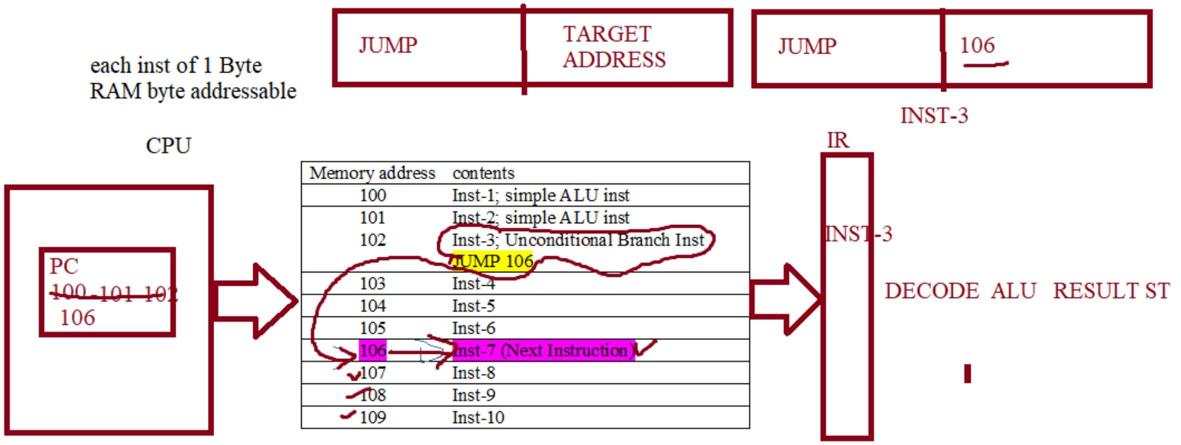
Memory address	contents
100	Inst-1; simple ALU inst
101	Inst-2; simple ALU inst
102	Inst-3; Unconditional Branch Inst JUMP 106
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7 (Next Instruction)
107	Inst-8
108	Inst-9
109	Inst-10

Datapath design for Fetch unit for Unconditional JUMP Instructions:

JUMP	Branch Target Address



each inst of 1 Byte
RAM byte addressable



Memory address	contents
100	Inst-1; simple ALU inst
101	Inst-2; simple ALU inst
102	Inst-3; Unconditional Branch Inst JUMP 106
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7 (Next Instruction)
107	Inst-8
108	Inst-9
109	Inst-10

- Unconditional branch CHANGE THE sequence of instruction as instructed in Instruction

Conditional Branch instructions

The general format of Conditional Branch instructions is as follows

Opcode	Reg-1	Reg-2	Relative Branch Target Address (offset: to be added to current PC to calculate address of next instruction)
BEQ	R1	R2	4

A program contains 10 instructions, each of one Byte and loaded into a byte addressable RAM starting at memory address 100.

Inst-3 is an conditional branch instruction:

BEQ R1, R2, 4

When the program is run, CPU fetches & executes Inst-1 followed by Inst-2 and Inst-3. Here Inst-3 is a conditional Branch instruction. Please note that when Inst-3 is decoded, the PC is incremented by one (PC = PC + 1 = 103) to point Inst-4 but Inst-3 is still under processing. Since Inst-3 is a conditional Branch Instruction, the CPU will evaluate the condition as indicted in the opcode. In this example instruction, the CPU will check whether the content of register R1 is equal to content of R2. In order to check the condition, CPU will read the contents of R1 and R2 and a subtraction operation can be performed at ALU. If the result of subtraction is found zero, then the condition is said to be evaluated TRUE. On the other hand, after subtraction, if the result is found non-zero, the condition is said to be evaluated FALSE.

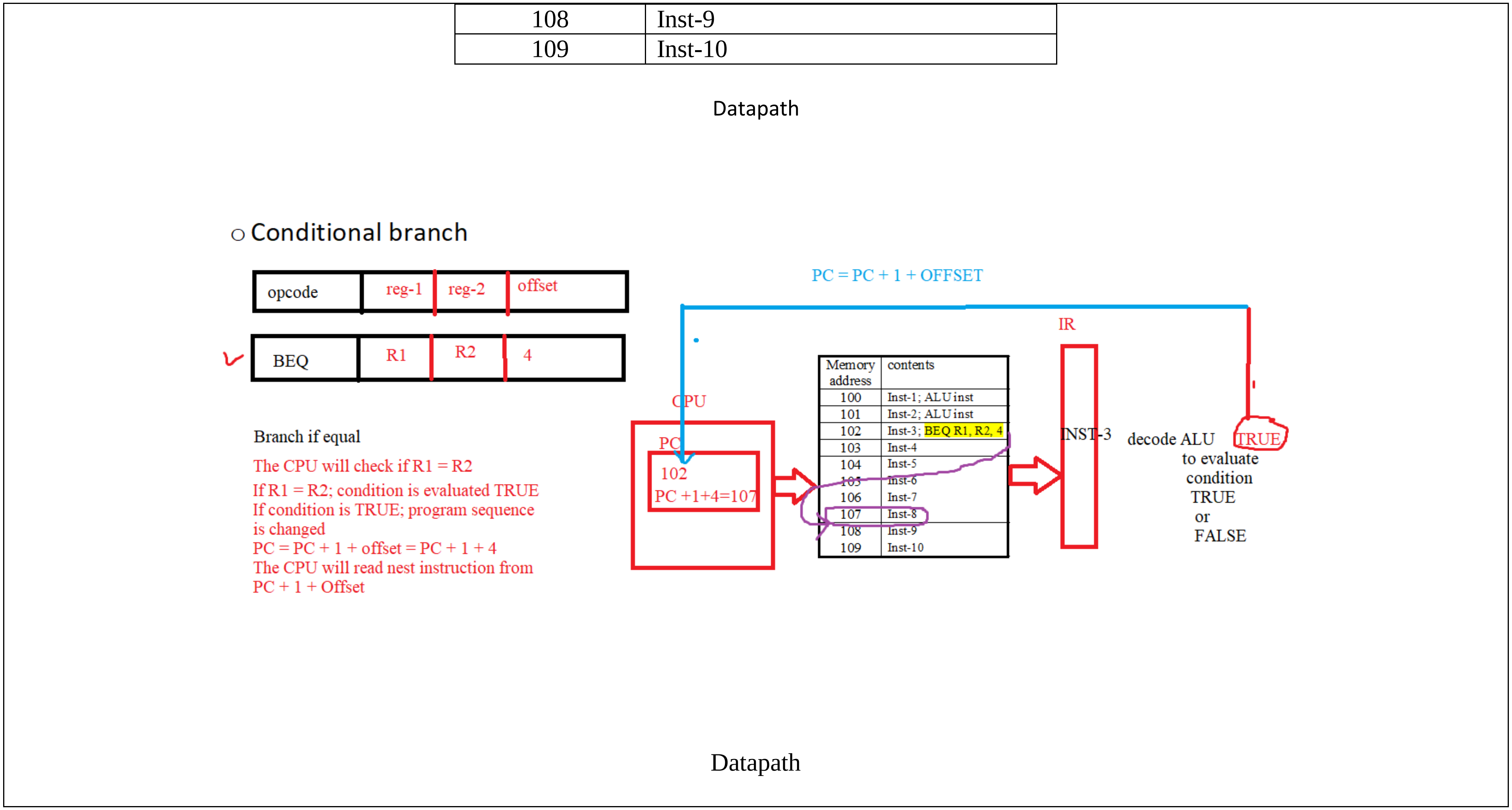
In any conditional branch instruction, first the condition is evaluated (TRUE/FALSE). If the condition is evaluated TRUE, the CPU will switch/change the sequence of instruction. If the condition is evaluated FALSE, the CPU will continue the sequence of instruction, means, Inst-4 will be fetched/processed next followed by Inst-5, Inst-6 so on.

If the condition is evaluated TRUE, the CPU will read next Instruction from a new location of RAM. The address of the location is called Branch Target address. As mentioned earlier, when Inst-3 was decoded, the program counter was incremented by one to point next instruction that was loaded in RAM. When the condition is evaluated TRUE, "Relative Branch Target Address" or the Offset value given in the instruction will be added to current content of PC to calculate the address of next Instruction.

PC = PC + 1 + Offset
Here PC = 102 + 1 + 4 = 107

Now the CPU will switch/Jump to memory address 107 (Target Address) to read next instruction. So after processing Inst-3, the CPU changes the sequence of instructions, that means it does not process Inst-4, Inst-5 rather it Jumps to Inst-8. The CPU will read Inst-8 and process it. Moreover the CPU will continue to process following instruction in sequence from memory address 106 until there appears another conditional instruction.

Memory address	contents
100	Inst-1; simple ALU inst
101	Inst-2; simple ALU inst
102	Inst-3; Conditional Branch Inst BEQ R1, R2, 4 (condition evaluated TRUE)
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8



108	Inst-9
109	Inst-10

Datapath

○ Conditional branch

opcode	reg-1	reg-2	offset
BEQ	R1	R2	4

Branch if equal

The CPU will check if R1 = R2

If R1 = R2; condition is evaluated TRUE

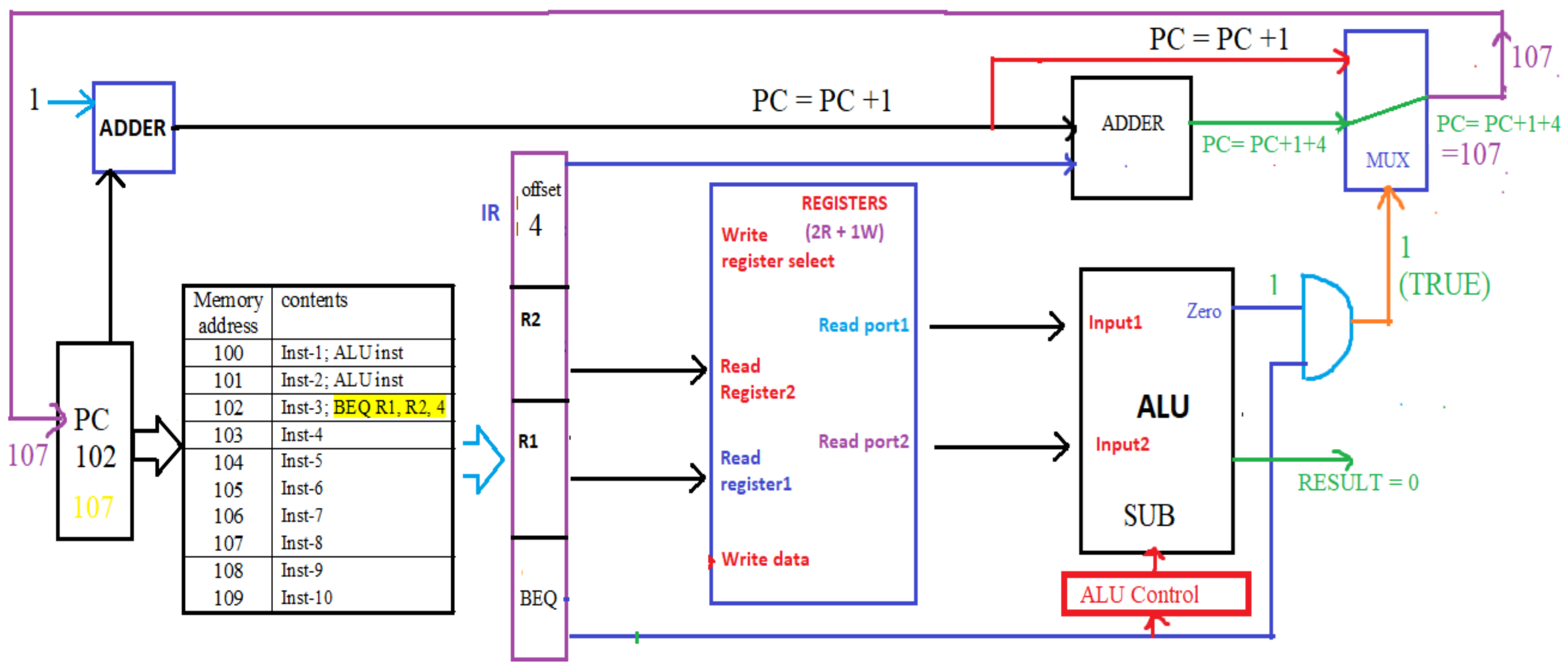
If condition is TRUE; program sequence is changed

PC = PC + 1 + offset = PC + 1 + 4

The CPU will read nest instruction from PC + 1 + Offset

Memory address	contents
100	Inst-1; ALU inst
101	Inst-2; ALU inst
102	Inst-3; BEQ R1, R2, 4
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

Datapath



Memory address	contents
100	Inst-1; ALU inst
101	Inst-2; ALU inst
102	Inst-3; BEQ R1, R2, 4
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

In case the condition ifs evaluated FALSE, the CPU will follow the current sequence, it means that the current content of PC (103) will point the next instruction. So CPU will fetch and process Inst-4 followed by Inst-5, Inst-6 so on.

Memory address	contents
100	Inst-1; simple ALU inst
101	Inst-2; simple ALU inst
102	Inst-3; Conditional Branch Inst BEQ R1, R2, 4 (condition evaluated FALSE)
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

Datapath

○ Conditional branch

opcode	reg-1	reg-2	offset
BEQ	R1	R2	4

Branch if equal

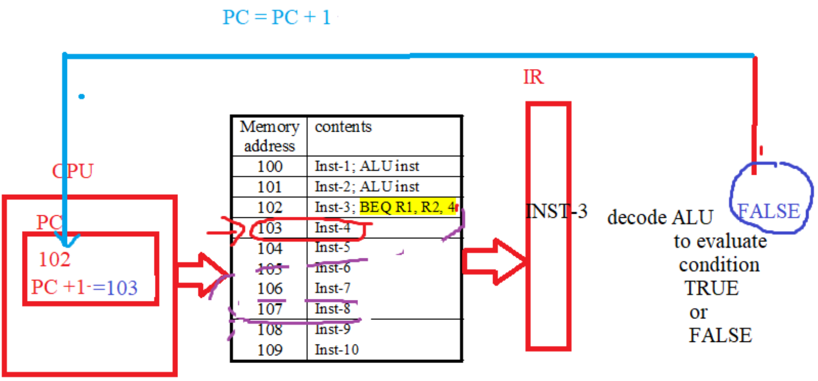
The CPU will check if R1 = R2

If R1 = R2; condition is evaluated TRUE

If condition is TRUE; program sequence is changed

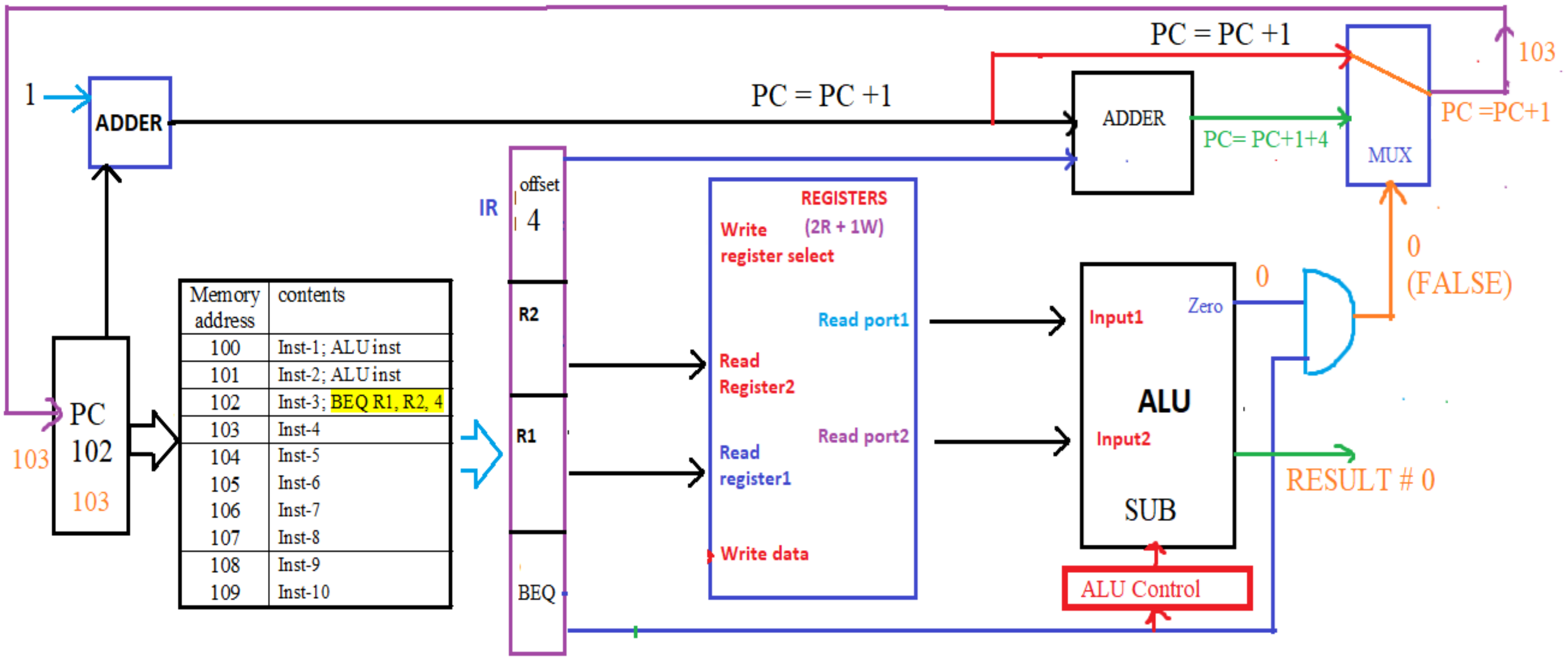
PC = PC + 1 + offset = PC + 1 + 4

The CPU will read nest instruction from PC + 1 + Offset



Memory address	contents
100	Inst-1; ALU inst
101	Inst-2; ALU inst
102	Inst-3; BEQ R1, R2, 4
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

Datapath



Memory address	contents
100	Inst-1; ALU inst
101	Inst-2; ALU inst
102	Inst-3; BEQ R1, R2, 4
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

Branch Instructions

- ○ Unconditional branch
- ○ Conditional branch
  - ▪ Branch IF condition is True

- If condition is False….continue

Another example:

Assume, a program contains 100 instructions, each of 32 bits (4 Bytes) and loaded into a byte addressable RAM starting at memory address 100. Each Instruction (Machine Code) would require 4 addressable locations (shown below) and Program Counter (PC) would be incremented by 4 to point next instruction once an instruction is fetched/executed.

When the program is run, CPU fetches & executes Inst-1 followed by Inst-2. Here Inst-2 is a conditional Branch instruction.

Here Inst-2 is a conditional Branch Instruction : BEQ R1, R2, 96

Please note that when Inst-2 is decoded, the PC is incremented by 4 (PC = PC + 4 = 104+4 = 108) to point Inst-3 but Inst-2 is still under processing. Since Inst-2 is a conditional Branch Instruction, the CPU will evaluate the condition as indicted in the opcode. In this example instruction, the CPU will check whether the content of register R1 is equal to content of R2. In order to check the condition, CPU will read the contents of R1 and R2 and a subtraction operation can be performed at ALU. If the result of subtraction is found zero, then the condition is said to be evaluated TRUE. On the other hand, after subtraction, if the result is found non-zero, the condition is said to be evaluated FALSE.

In any conditional branch instruction, first the condition is evaluated (TRUE/FALSE). If the condition is evaluated TRUE, the CPU will switch/change the sequence of instruction. If the condition is evaluated FALSE, the CPU will continue the sequence of instruction, means, Inst-3 will be fetched/processed next followed by Inst-4, Inst-5 so on.

If the condition is evaluated TRUE, the CPU will read next Instruction from a new location of RAM. The address of the location is called Branch Target address. As mentioned earlier, when Inst-2 was decoded, the program counter was incremented by 4 to point next instruction that was loaded in RAM. When the condition is evaluated TRUE, “Relative Branch Target Address” or the Offset value given in the instruction will be added to current content of PC to calculate the address of next Instruction.

PC = PC + 4 + Offset

Here PC = 104 + 4 + 96 = 204

Now the CPU will switch/Jump to memory address 204 (Target Address) to read next instruction. So after processing Inst-2, the CPU changes the sequence of instructions that means it does not process Inst-3, Inst-4 rather it Jumps to Inst-25. The CPU will read Inst-25 and process it. Moreover the CPU will continue to process following instruction in sequence from memory address 204 until there appears another conditional instruction.

Memory address	Contents	Types of Inst
100	Inst-1 lower 8 bits	ALU
101	Inst-1 next 8 bits	
102	Inst-1 next 8 bits	
103	Inst-1 next 8 bits	
104	Inst-2 (BEQ R1, R2, 96)	Branch Inst (Conditional) PC = PC + 4 = 108
105		
106		
107		
108	Inst-3 (IF branch condition is FALSE)	
109		
--		
--		
--		
--		
204	Inst-25 (IF branch condition is TRUE)	PC = PC + 4 + 96 = 204
205		
206		
207		
208	Inst-26	

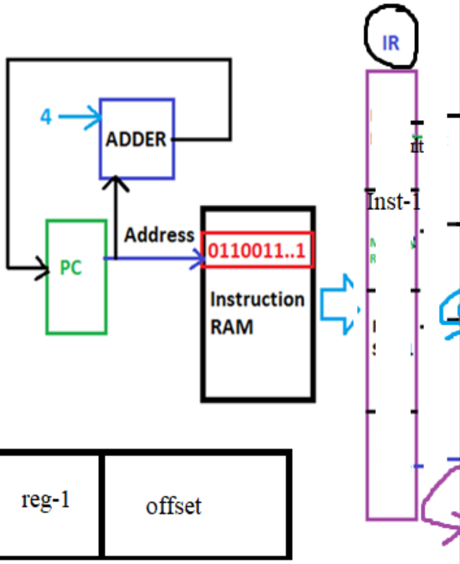
Each Inst is of 32 bits (4 bytes) and RAM is byte addressable
Once an Inst is Fetched, PC is incremented by 4 to point next Inst; PC = PC + 4

BEQ R1, R2, 96 ; Branch if R1 = R2;
CPU will check whether R1 = R2
If Branch condition is evaluated TRUE then CPU will Jump to memory address
PC = PC + 4 + 96 = 104 + 4 + 96 = 204 to read next Instruction (Inst-25)

If R1 is Not equal to R2, Branch condition is evaluated FALSE
Then continue next instruction that appears in the program, means CPU will read next instruction from PC = PC + 4

At th Beginning
PC = 100 (Inst-1)
Inst-1 is Fetched
PC = 100 + 4 =104
Inst-2: Conditional Branch
Inst-2 Fetched
PC = PC + 4 = 108
Condition False Next Inst-3
PC = 104+4= 108 Inst-3

Condition TRUE; next (Inst-25)
PC = PC + 4 + 96 = 204
PC = 108 + 4 + 96 = 204

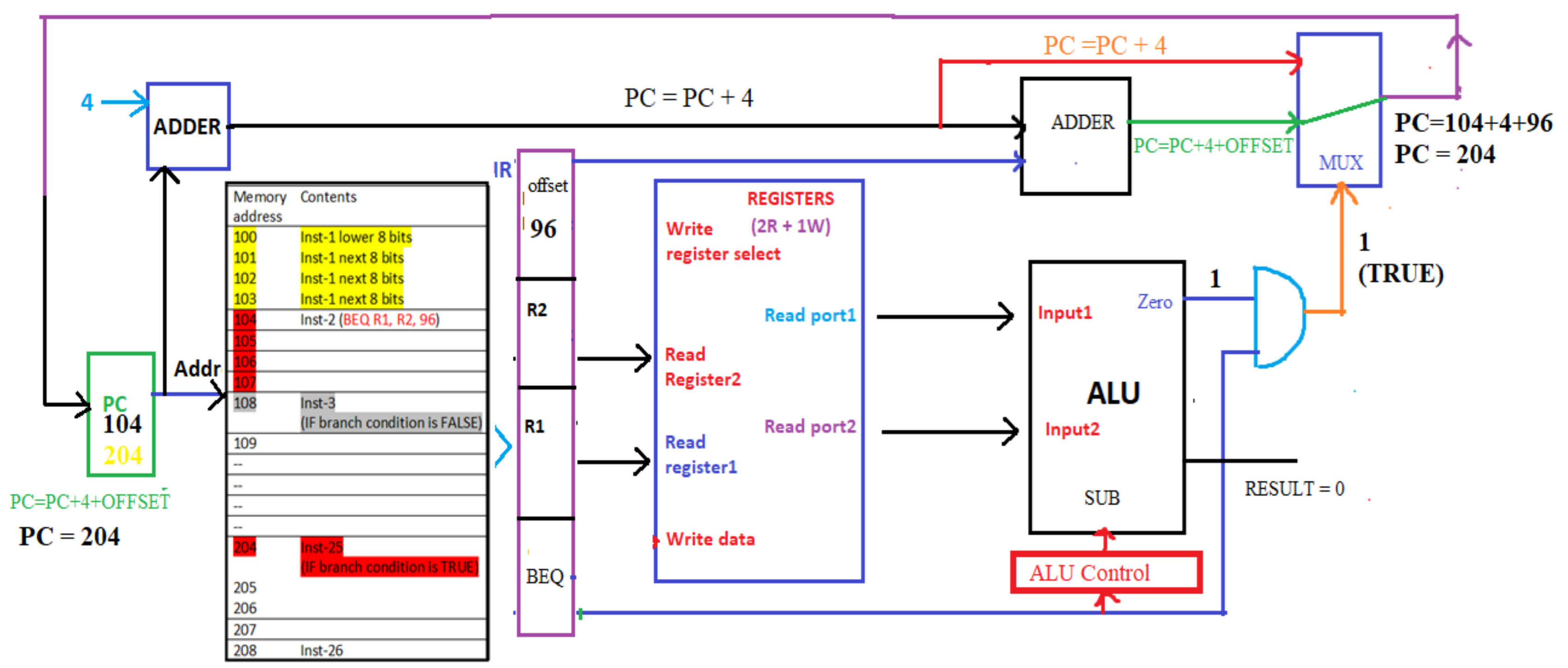


Memory address	Contents	Types of Inst
100	Inst-1 lower 8 bits	ALU
101	Inst-1 next 8 bits	
102	Inst-1 next 8 bits	
103	Inst-1 next 8 bits	
104	Inst-2 (BEQ R1, R2, 96) FALSE TRUE	Branch Inst (Condition PC = PC + 4 = 108
105		
106		
107		
108	Inst-3 (IF branch condition is FALSE)	
109		
--		
--		
--		
--		
204	Inst-25 (IF branch condition is TRUE)	PC = PC + 4 + 96 = 204
205		
206		
207		
208	Inst-26	

opcode	reg-1	reg-1	offset
BEQ	R1	R2	96

Datapath design

Datapath design: Condition evaluated TRUE

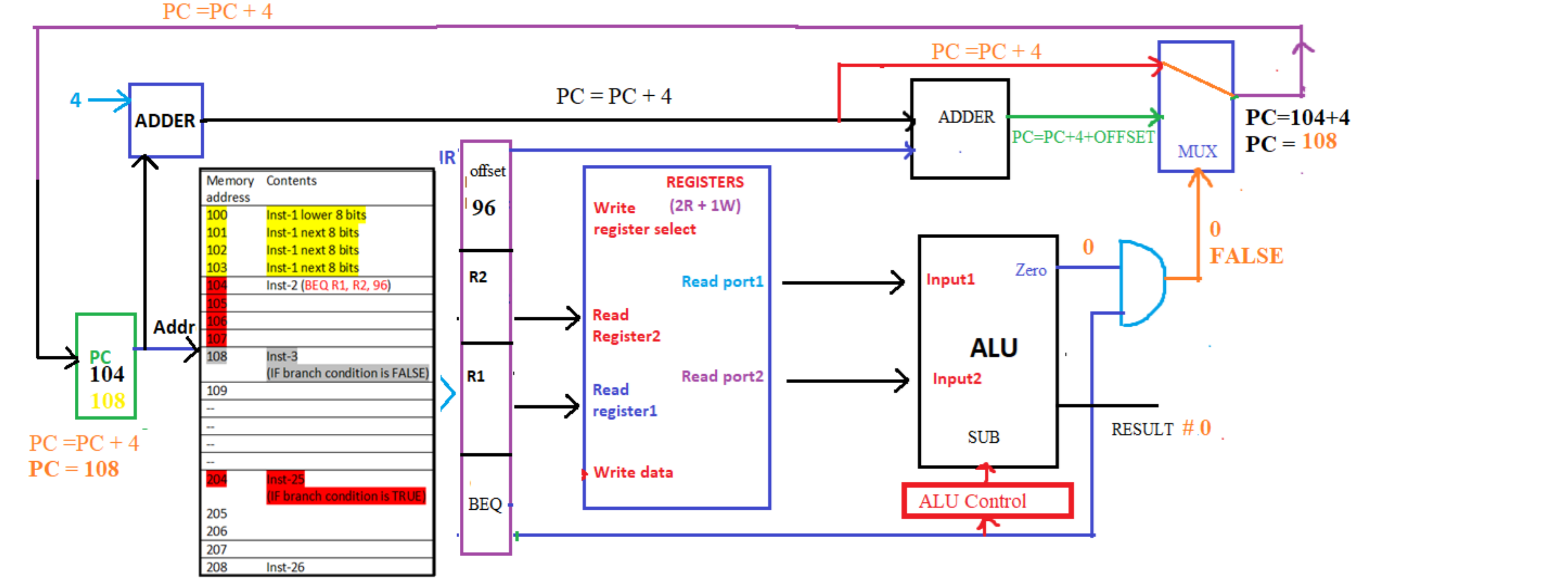


Memory address	Contents
100	Inst-1 lower 8 bits
101	Inst-1 next 8 bits
102	Inst-1 next 8 bits
103	Inst-1 next 8 bits
104	Inst-2 (BEQ R1, R2, 96)
105	
106	
107	
108	Inst-3 (IF branch condition is FALSE)
109	
--	
--	
--	
--	
204	Inst-25 (IF branch condition is TRUE)
205	
206	
207	
208	Inst-26

--

Datapath design: Condition evaluated FALSE

In case the condition is evaluated FALSE, the CPU will follow the current sequence, it means that the current content of PC (108) will point the next instruction. So CPU will fetch and process Inst-3 followed by Inst-4, Inst-5 so on.



--

What are branch instruction? Explain, how a branch instruction works with an example and suitable diagrams.

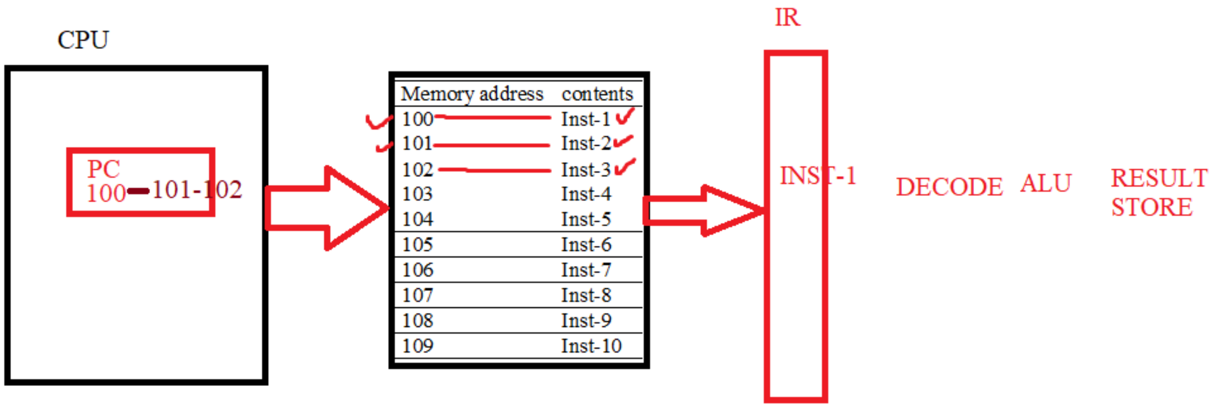
The branch instructions are used to change the sequence of instruction execution. To solve problems or to implement different types of algorithms, users/programmers use branch instructions to change the sequence of instruction execution in a program.

A program contains 10 instructions, each of one Byte and loaded into a byte addressable RAM starting at memory address 100.

If there is no branch instruction in the program, the CPU will fetch & process instructions in sequence as it appears in the program and loaded in memory location. It means, the CPU will fetch & process Inst-1, followed by Inst-2, Inst-3, Inst-4 so on and continue until the end of the list.

Memory address	contents
100	Inst-1
101	Inst-2
102	Inst-3
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

each inst of 1 Byte
RAM byte addressable



Memory address	contents
100	Inst-1 ✓
101	Inst-2 ✓
102	Inst-3 ✓
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

Branch instructions are divided into two categories : (i) Unconditional Branch and (ii) Conditional branch instructions.

The general format of Unconditional Branch instructions is as follows

Opcode	Target Address (Memory address to read instruction)
JUMP	106

A program contains 10 instructions, each of one Byte and loaded into a byte addressable RAM starting at memory address 100.

Inst-3 is an unconditional branch instruction:

JUMP 106

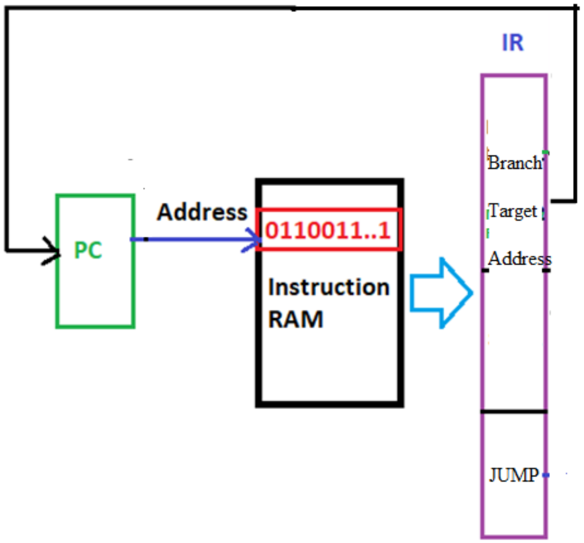
When the program is run, CPU fetches & executes Inst-1 followed by Inst-2 and Inst-3. Here Inst-3 is an unconditional Jump instruction and it instructs the CPU to Jump to memory address 106 (Target Address) to read next instruction. So after processing Inst-3, the CPU changes the sequence of instructions, that means it does not process Inst-4, Inst-5 rather it Jumps to Inst-7. The CPU will read Inst-7 and process it. Moreover the CPU will continue to process following instruction in sequence from memory address 106 until there appears another conditional instruction.

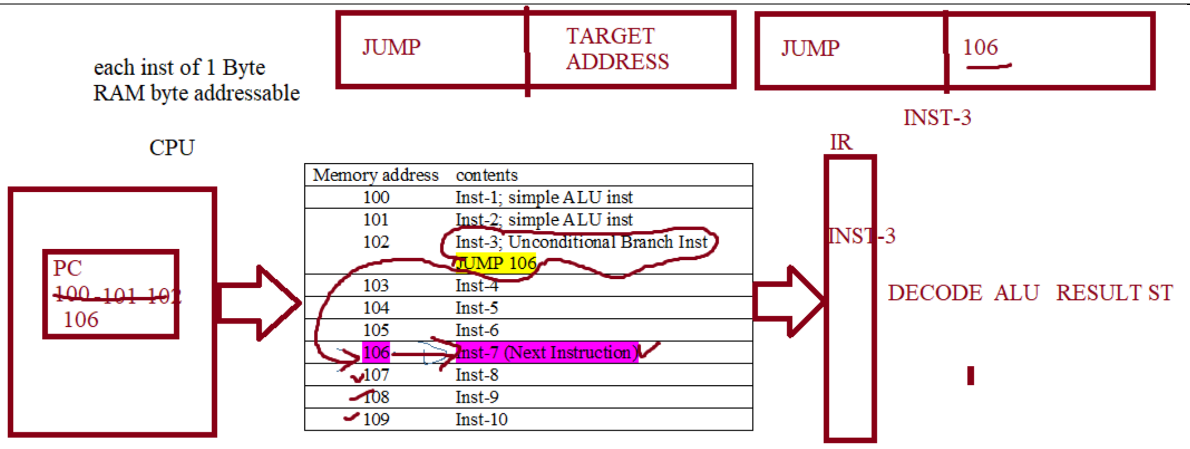
Memory address	contents
100	Inst-1; simple ALU inst
101	Inst-2; simple ALU inst
102	Inst-3; Unconditional Branch Inst JUMP 106
103	Inst-4

104	Inst-5
105	Inst-6
106	Inst-7 (Next Instruction)
107	Inst-8
108	Inst-9
109	Inst-10

Datapath design for Fetch unit for Unconditional JUMP Instructions:

JUMP	Branch Target Address





Memory address	contents
100	Inst-1; simple ALU inst
101	Inst-2; simple ALU inst
102	Inst-3; Unconditional Branch Inst JUMP 106
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7 (Next Instruction)
107	Inst-8
108	Inst-9
109	Inst-10

○ Unconditional branch CHANGE THE sequence of instruction as instructed in Instruction

Conditional Branch instructions

The general format of Conditional Branch instructions is as follows

Opcode	Reg-1	Reg-2	Relative Branch Target Address (offset: to be added to current PC to calculate address of next instruction)
BEQ	R1	R2	4

A program contains 10 instructions, each of one Byte and loaded into a byte addressable RAM starting at memory address 100.

Inst-3 is an conditional branch instruction:

BEQ R1, R2, 4

When the program is run, CPU fetches & executes Inst-1 followed by Inst-2 and Inst-3. Here Inst-3 is a conditional Branch instruction. Please note that when Inst-3 is decoded, the PC is incremented by one (PC = PC + 1 = 103) to point Inst-4 but Inst-3 is still under processing. Since Inst-3 is a conditional Branch Instruction, the CPU will evaluate the condition as indicted in the opcode. In this example instruction, the CPU will check whether the content of register R1 is equal to content of R2. In order to check the condition, CPU will read the contents of R1 and R2 and a subtraction operation can be performed at ALU. If the result of subtraction is found zero, then the condition is said to be evaluated TRUE. On the other hand, after subtraction, if the result is found non-zero, the condition is said to be evaluated FALSE.

In any conditional branch instruction, first the condition is evaluated (TRUE/FALSE). If the condition is evaluated TRUE, the CPU will switch/change the sequence of instruction. If the condition is evaluated FALSE, the CPU will continue the sequence of instruction, means, Inst-4 will be fetched/processed next followed by Inst-5, Inst-6 so on.

If the condition is evaluated TRUE, the CPU will read next Instruction from a new location of RAM. The address of the location is called Branch Target address. As mentioned earlier, when Inst-3 was decoded, the program counter was incremented by one to point next instruction that was loaded in RAM. When the condition is evaluated TRUE, "Relative Branch Target Address" or the Offset value given in the instruction will be added to current content of PC to calculate the address of next Instruction.

PC = PC + 1 + Offset
Here PC = 102 + 1 + 4 = 107

Now the CPU will switch/Jump to memory address 107 (Target Address) to read next instruction. So after processing Inst-3, the CPU changes the sequence of instructions, that means it does not process Inst-4, Inst-5 rather it Jumps to Inst-8. The CPU will read Inst-8 and process it. Moreover the CPU will continue to process following instruction in sequence from memory address 106 until there appears another conditional instruction.

Memory address	contents
100	Inst-1; simple ALU inst
101	Inst-2; simple ALU inst
102	Inst-3; Conditional Branch Inst BEQ R1, R2, 4 (condition evaluated TRUE)
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

Datapath

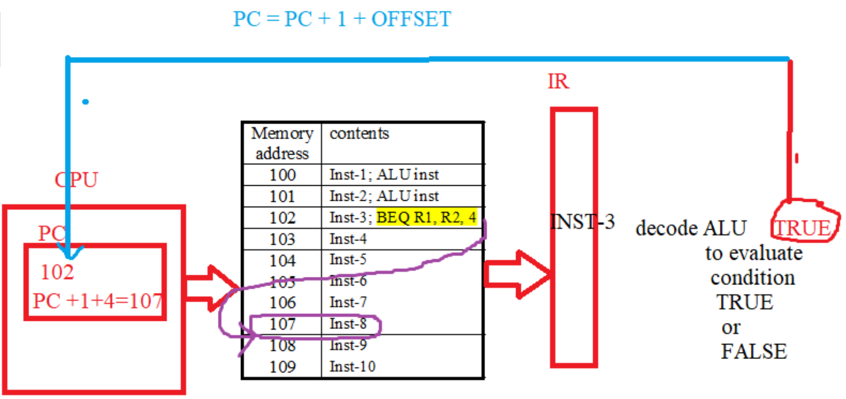
○ Conditional branch

opcode	reg-1	reg-2	offset
BEQ	R1	R2	4

Branch if equal

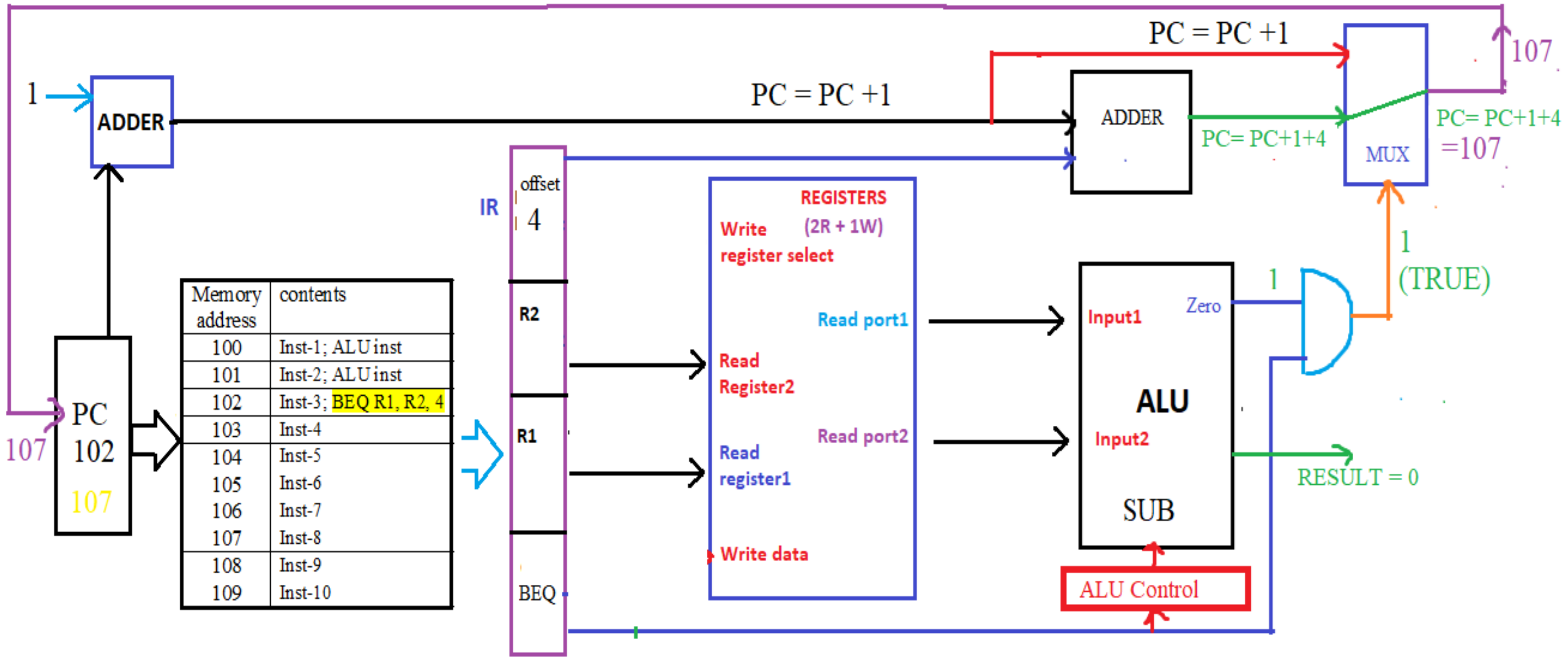
The CPU will check if R1 = R2

If R1 = R2; condition is evaluated TRUE
If condition is TRUE; program sequence is changed
PC = PC + 1 + offset = PC + 1 + 4
The CPU will read nest instruction from PC + 1 + Offset



Memory address	contents
100	Inst-1; ALU inst
101	Inst-2; ALU inst
102	Inst-3; BEQ R1, R2, 4
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

Datapath



Memory address	contents
100	Inst-1; ALU inst
101	Inst-2; ALU inst
102	Inst-3; BEQ R1, R2, 4
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

In case the condition ifs evaluated FALSE, the CPU will follow the current sequence, it means that the current content of PC (103) will point the next instruction. So CPU will fetch and process Inst-4 followed by Inst-5, Inst-6 so on.

Memory address	contents
100	Inst-1; simple ALU inst
101	Inst-2; simple ALU inst
102	Inst-3; Conditional Branch Inst BEQ R1, R2, 4 (condition evaluated FALSE)
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

Datapath

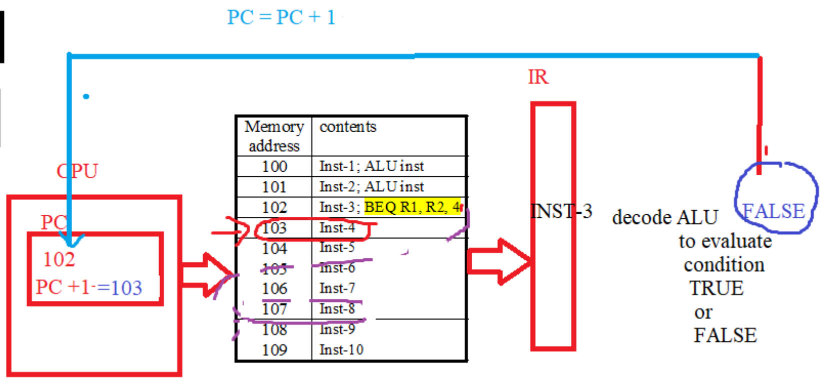
○ Conditional branch

opcode	reg-1	reg-2	offset
BEQ	R1	R2	4

Branch if equal

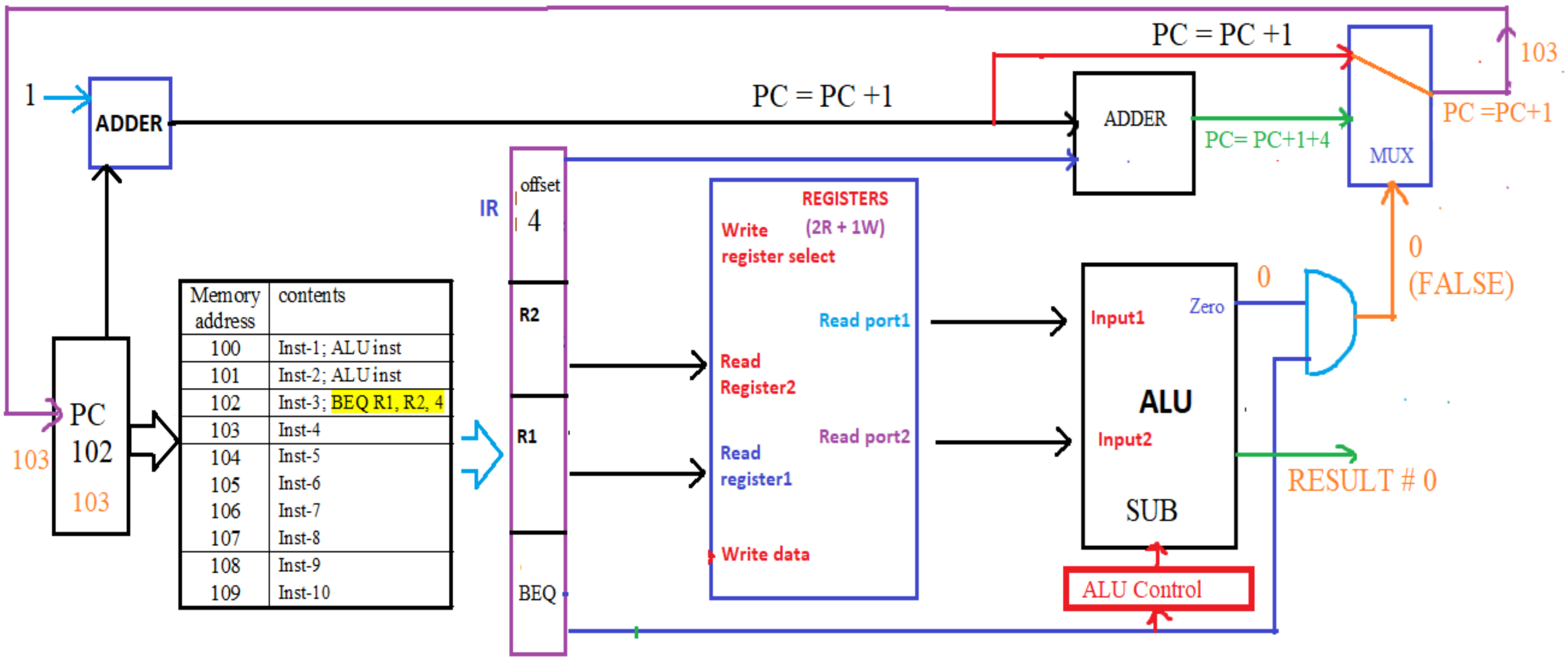
The CPU will check if R1 = R2

If R1 = R2; condition is evaluated TRUE
If condition is TRUE; program sequence is changed
PC = PC + 1 + offset = PC + 1 + 4
The CPU will read nest instruction from PC + 1 + Offset



Memory address	contents
100	Inst-1; ALU inst
101	Inst-2; ALU inst
102	Inst-3; BEQ R1, R2, 4
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

Datapath



Memory address	contents
100	Inst-1; ALU inst
101	Inst-2; ALU inst
102	Inst-3; BEQ R1, R2, 4
103	Inst-4
104	Inst-5
105	Inst-6
106	Inst-7
107	Inst-8
108	Inst-9
109	Inst-10

Branch Instructions

- Unconditional branch
- Conditional branch
    - Branch IF condition is True
    - If condition is False….continue

Another example:

Assume, a program contains 100 instructions, each of 32 bits (4 Bytes) and loaded into a byte addressable RAM starting at memory address 100. Each Instruction (Machine Code) would require 4 addressable locations (shown below) and Program Counter (PC) would be incremented by 4 to point next instruction once an instruction is fetched/executed.

When the program is run, CPU fetches & executes Inst-1 followed by Inst-2. Here Inst-2 is a conditional Branch instruction.

Here Inst-2 is a conditional Branch Instruction : BEQ R1, R2, 96

Please note that when Inst-2 is decoded, the PC is incremented by 4 (PC = PC + 4 = 104+4 = 108) to point Inst-3 but Inst-2 is still under processing. Since Inst-2 is a conditional Branch Instruction, the CPU will evaluate the condition as indicted in the opcode. In this example instruction, the CPU will check whether the content of register R1 is equal to content of R2. In order to check the condition, CPU will read the contents of R1 and R2 and a subtraction operation can be performed at ALU. If the result of subtraction is found zero, then the condition is said to be evaluated TRUE. On the other hand, after subtraction, if the result is found non-zero, the condition is said to be evaluated FALSE.

In any conditional branch instruction, first the condition is evaluated (TRUE/FALSE). If the condition is evaluated TRUE, the CPU will switch/change the sequence of instruction. If the condition is evaluated FALSE, the CPU will continue the sequence of instruction, means, Inst-3 will be fetched/processed next followed by Inst-4, Inst-5 so on.

If the condition is evaluated TRUE, the CPU will read next Instruction from a new location of RAM. The address of the location is called Branch Target address. As mentioned earlier, when Inst-2 was decoded, the program counter was incremented by 4 to point next instruction that was loaded in RAM. When the condition is evaluated TRUE, "Relative Branch Target Address" or the Offset value given in the instruction will be added to current content of PC to calculate the address of next Instruction.

PC = PC + 4 + Offset

Here PC = 104 + 4 + 96 = 204

Now the CPU will switch/Jump to memory address 204 (Target Address) to read next instruction. So after processing Inst-2, the CPU changes the sequence of instructions that means it does not process Inst-3, Inst-4 rather it Jumps to Inst-25. The CPU will read Inst-25 and process it. Moreover the CPU will continue to process following instruction in sequence from memory address 204 until there appears another conditional instruction.

Memory address	Contents	Types of Inst
100	Inst-1 lower 8 bits	ALU
101	Inst-1 next 8 bits	
102	Inst-1 next 8 bits	
103	Inst-1 next 8 bits	
104	Inst-2 (BEQ R1, R2, 96)	Branch Inst (Conditional) PC = PC + 4 = 108
105		
106		
107		
108	Inst-3 (IF branch condition is FALSE)	
109		
--		
--		

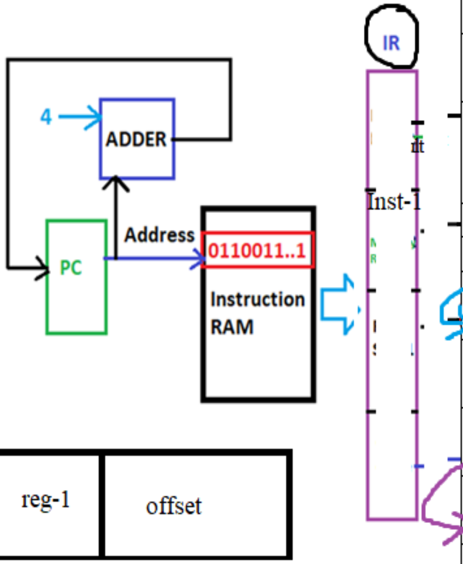
--		
--		
204	Inst-25 (IF branch condition is TRUE)	PC = PC + 4 + 96 = 204
205		
206		
207		
208	Inst-26	

Each Inst is of 32 bits (4 bytes) and RAM is byte addressable
Once an Inst is Fetched, PC is incremented by 4 to point next Inst; PC = PC + 4

BEQ R1, R2, 96 ; Branch if R1 = R2;
CPU will check whether R1 = R2
If Branch condition is evaluated TRUE then CPU will Jump to memory address
PC = PC + 4 + 96 = 104 + 4 + 96 = 204 to read next Instruction (Inst-25)

If R1 is Not equal to R2, Branch condition is evaluated FALSE
Then continue next instruction that appears in the program, means CPU will read next instruction from PC = PC + 4

At th Beginning
PC = 100 (Inst-1)
Inst-1 is Fetched
PC = 100 + 4 =104
Inst-2: Conditional Branch
Inst-2 Fetched
PC = PC + 4 = 108
Condition False Next Inst-3
PC = 104+4= 108 Inst-3
Condition TRUE; next (Inst-25)
PC = PC + 4 + 96 = 204
PC = 108 + 4 + 96 = 204



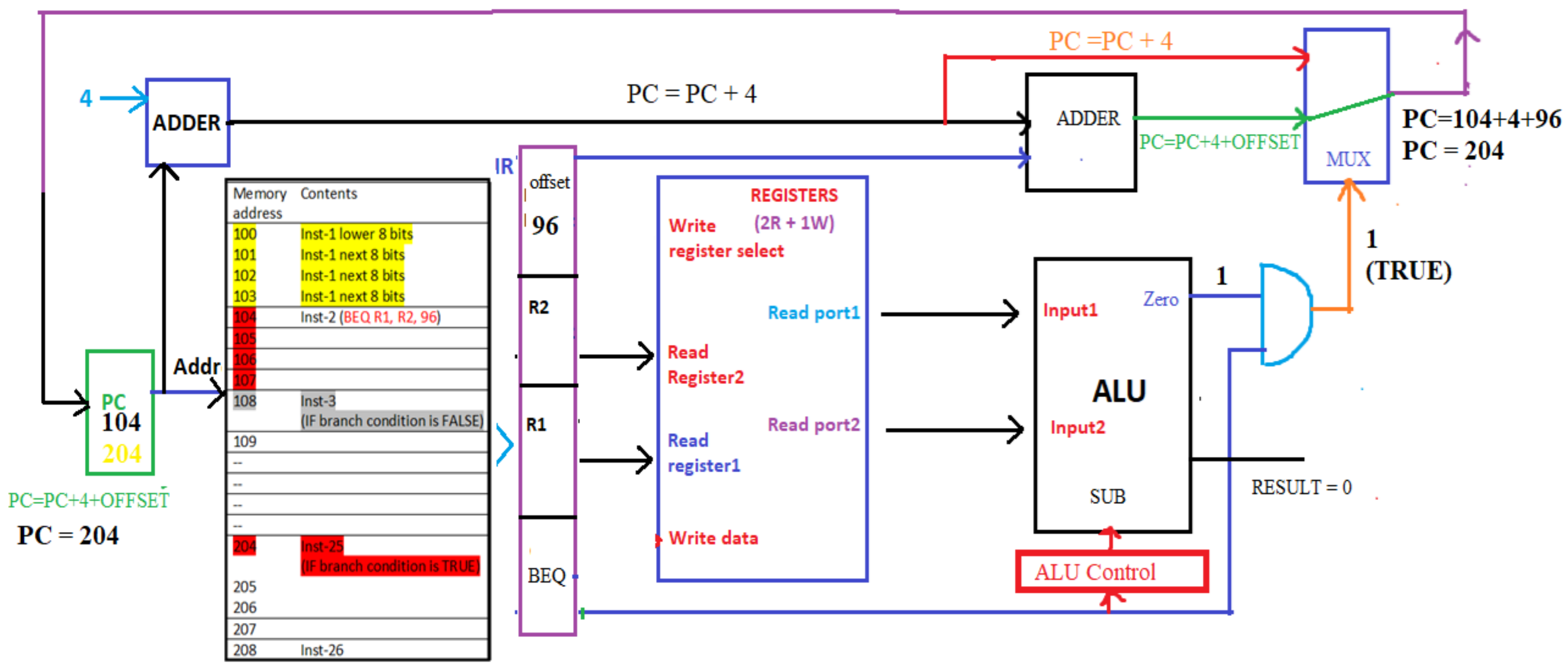
Memory address	Contents	Types of Inst
100	Inst-1 lower 8 bits	ALU
101	Inst-1 next 8 bits	
102	Inst-1 next 8 bits	
103	Inst-1 next 8 bits	
104	Inst-2 (BEQ R1, R2, 96) FALSE TRUE	Branch Inst (Condition PC = PC + 4 = 108
105		
106		
107		
108	Inst-3 (IF branch condition is FALSE)	
109		
--		
--		
--		
--		
204	Inst-25 (IF branch condition is TRUE)	PC = PC + 4 + 96 = 204
205		
206		
207		
208	Inst-26	

opcode	reg-1	reg-1	offset

BEQ	R1	R2	96

Datapath design

Datapath design: Condition evaluated TRUE

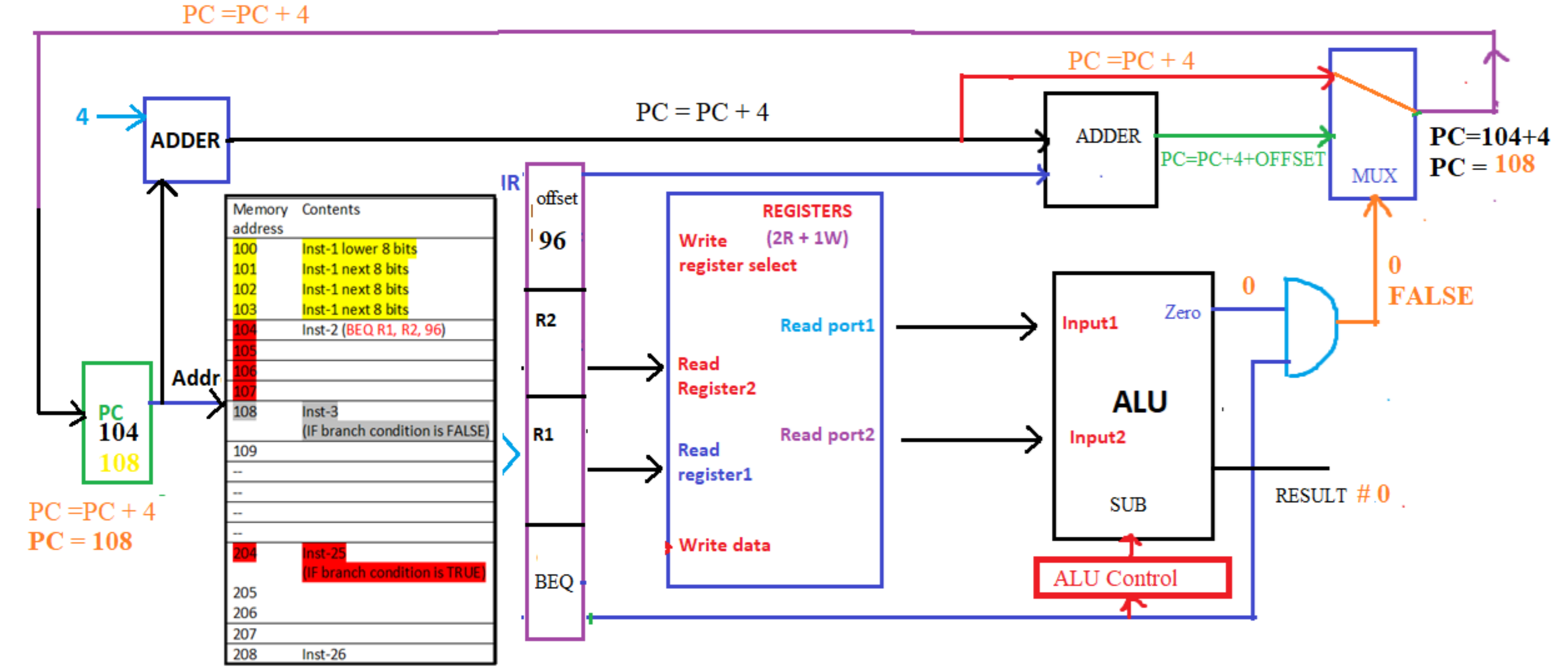


Memory address	Contents
100	Inst-1 lower 8 bits
101	Inst-1 next 8 bits
102	Inst-1 next 8 bits
103	Inst-1 next 8 bits
104	Inst-2 (BEQ R1, R2, 96)
105	
106	
107	
108	Inst-3 (IF branch condition is FALSE)
109	
--	
--	
--	
--	
204	Inst-25 (IF branch condition is TRUE)
205	
206	
207	
208	Inst-26

--

Datapath design: Condition evaluated FALSE

In case the condition is evaluated FALSE, the CPU will follow the current sequence, it means that the current content of PC (108) will point the next instruction. So CPU will fetch and process Inst-3 followed by Inst-4, Inst-5 so on.



Memory address	Contents
100	Inst-1 lower 8 bits
101	Inst-1 next 8 bits
102	Inst-1 next 8 bits
103	Inst-1 next 8 bits
104	Inst-2 (BEQ R1, R2, 96)
105	
106	
107	
108	Inst-3 (IF branch condition is FALSE)
109	
--	
--	
--	
--	
204	Inst-25 (IF branch condition is TRUE)
205	
206	
207	
208	Inst-26

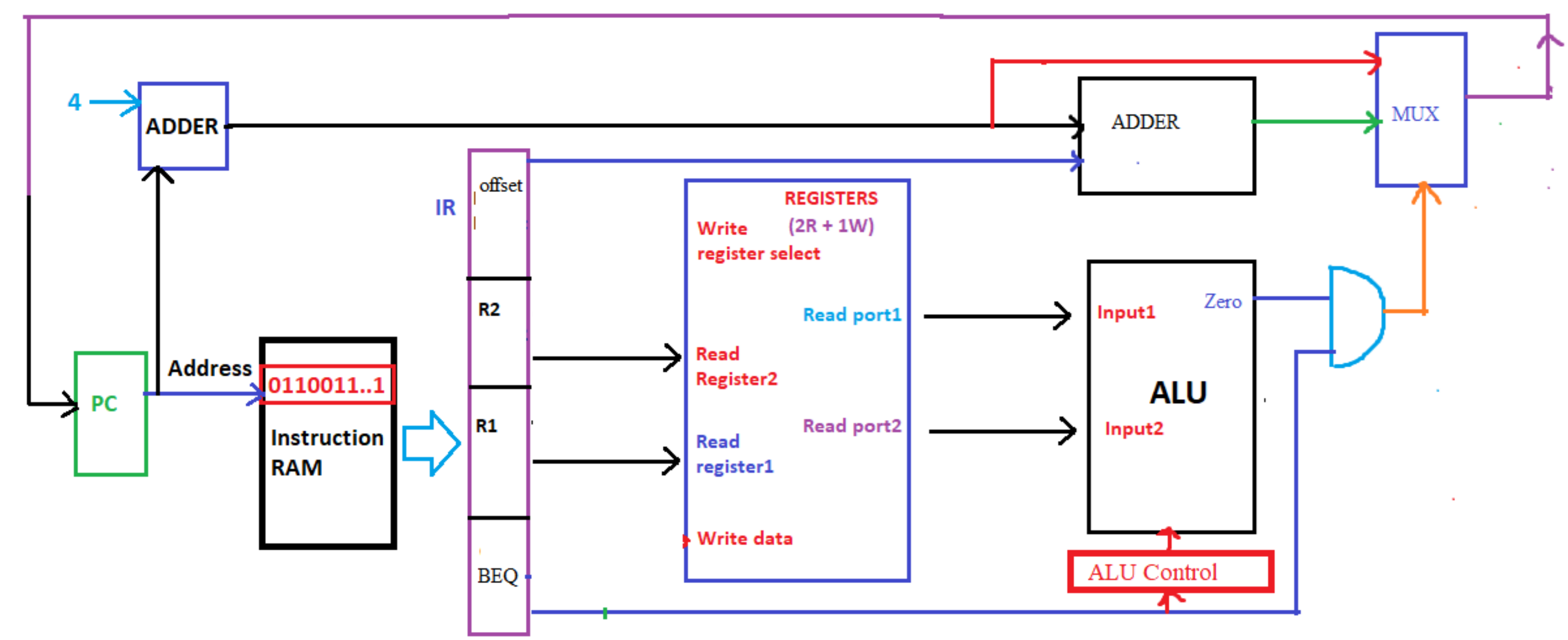
--

Design datapath for the Branch Instruction, given below

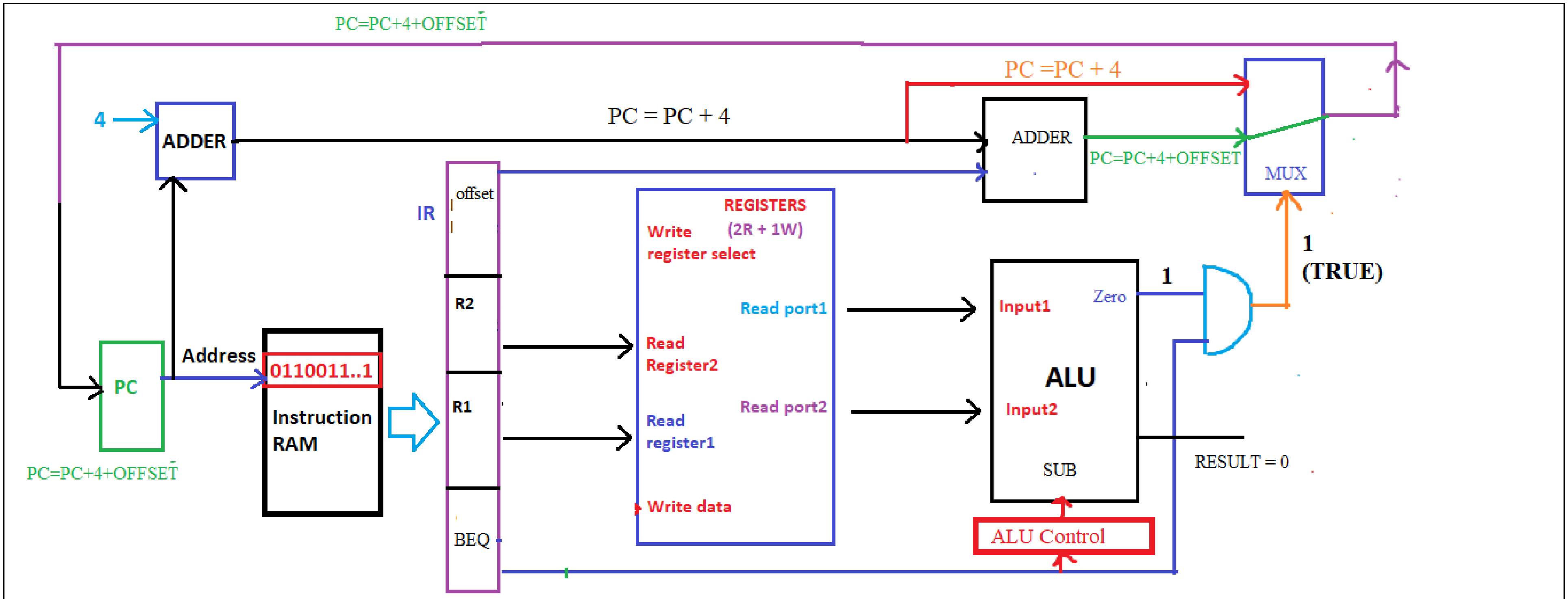
BEQ	R1	R2	Offset

Please note that, instructions are assumed to be 32 bits each and a byte addressable memory is used. So, PC is incremented by 4 to point next instruction once an instruction is fetched/executed.

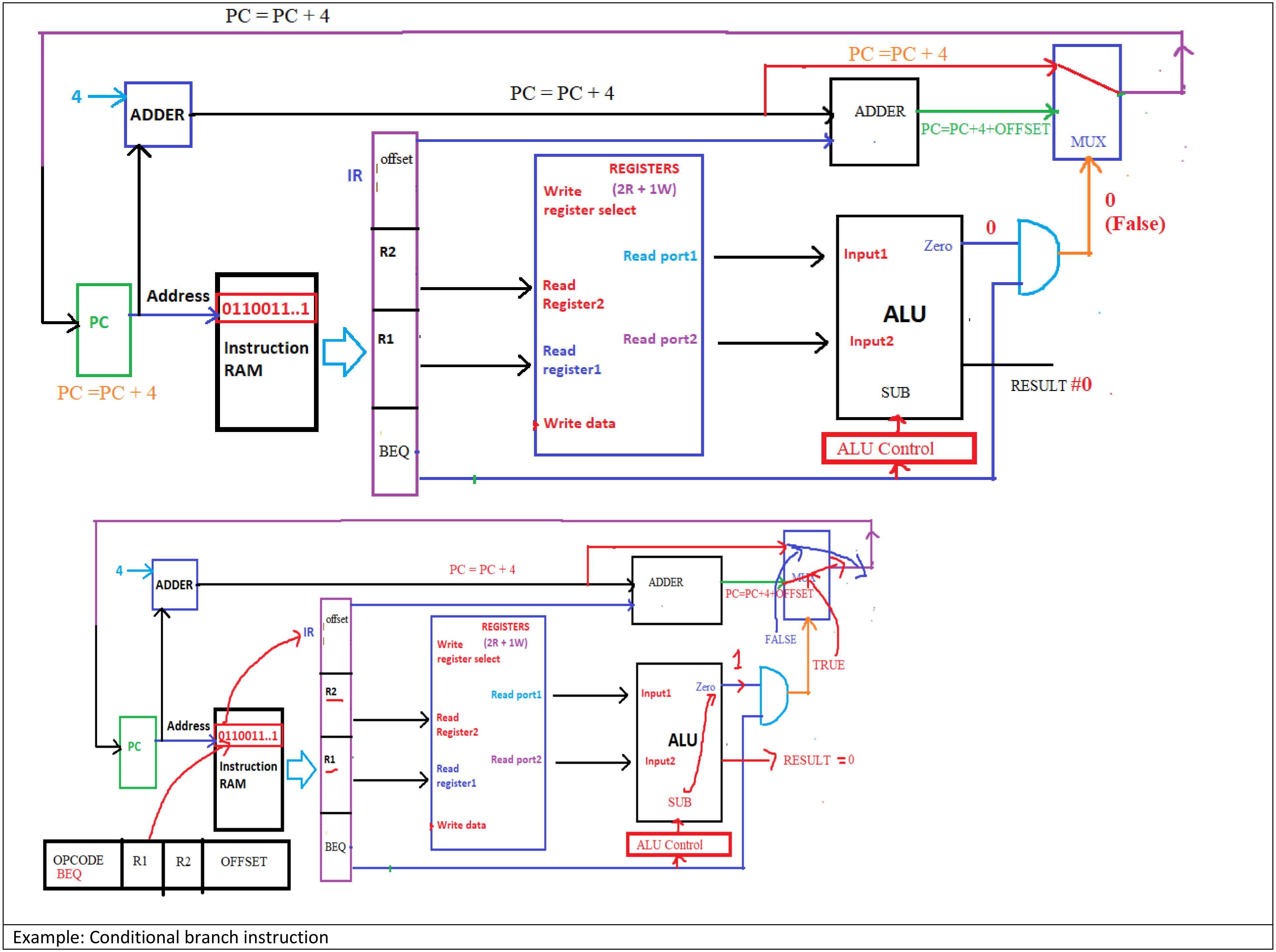
After executing the conditional branch Instruction: BEQ R1, R2, OFFSET, the CPU will read next instruction from PC = PC + 4 + Offset, if R1 = R2. On the other hand, CPU will read next instruction from PC = PC + 4, if R1 # R2. It is noted that PC is pointing conditional branch Instruction: BEQ R1, R2, OFFSET in the RAM. Moreover, PC = PC + 4 points the next instruction appears in program as well as stored next to conditional branch Instruction: BEQ R1, R2, OFFSET in the RAM. Furthermore, PC = PC + 4 + Offset points to a different instruction in RAM which does not lie in a sequence in RAM following conditional branch Instruction: BEQ R1, R2, OFFSET.



DATAPATH: if condition TRUE



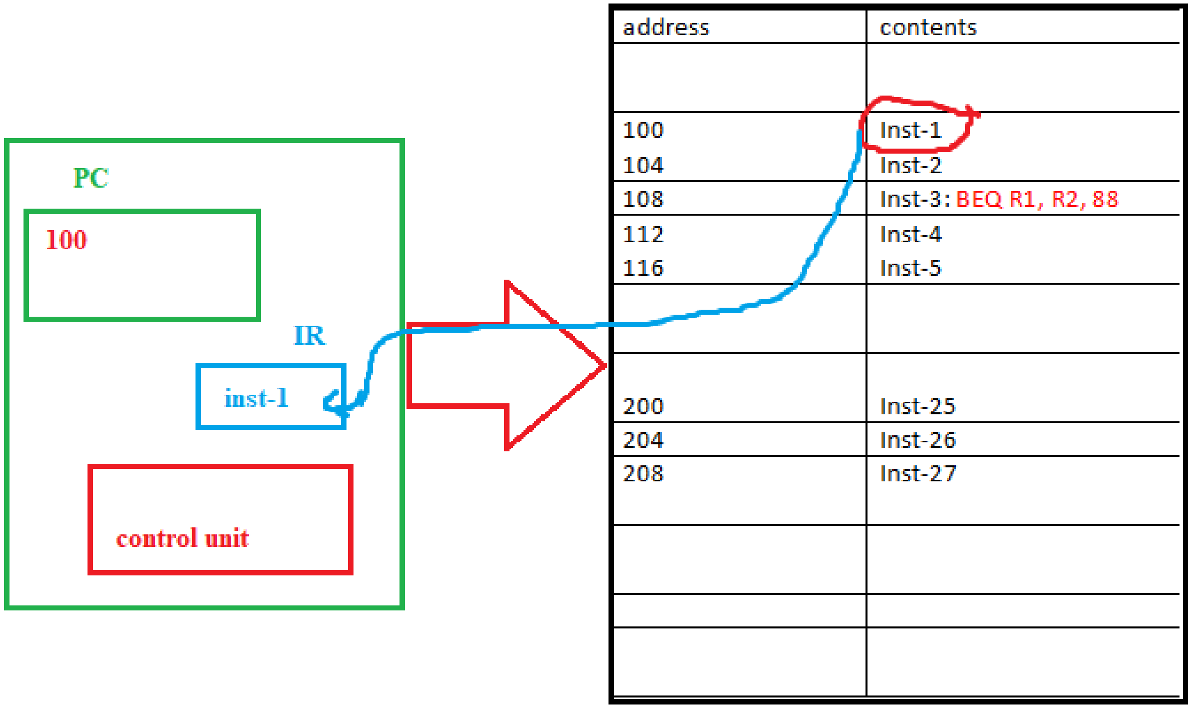
Datapath if condition False



Example: Conditional branch instruction

PC = 100
IR = Inst-1 (Fetch)
PC=100+4=104
Inst-1 decode
Inst-1 Execution
IR = Inst-2 (fetch)
PC=104+4=108
Inst-2 decode
Inst-2 Execution
IR=Inst-3 (fetch)
PC = 108 +4=112
Inst-3 decode
Inst-3 Execution
    Condition: TRUE
    PC = 112+88=200
    IR = Inst-25(Fetch)
    PC = 200+4=204

    Condition FALSE
    IR = Inst-4 (Fetch)
    PC = 112 + 4 =116
    Inst-4 decode
    Inst-4 Execution

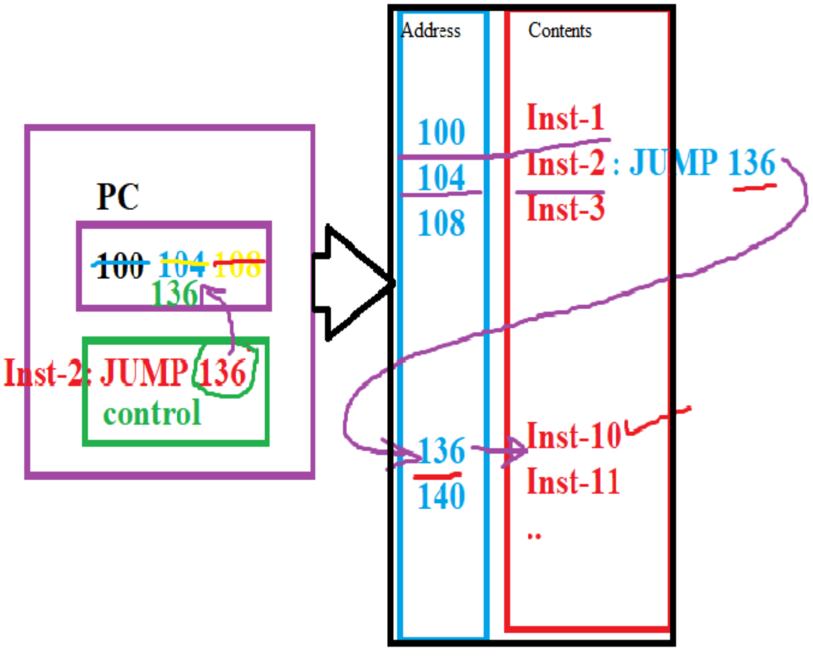


address	contents
100	Inst-1
104	Inst-2
108	Inst-3: BEQ R1, R2, 88
112	Inst-4
116	Inst-5
200	Inst-25
204	Inst-26
208	Inst-27

JUMP	Target address

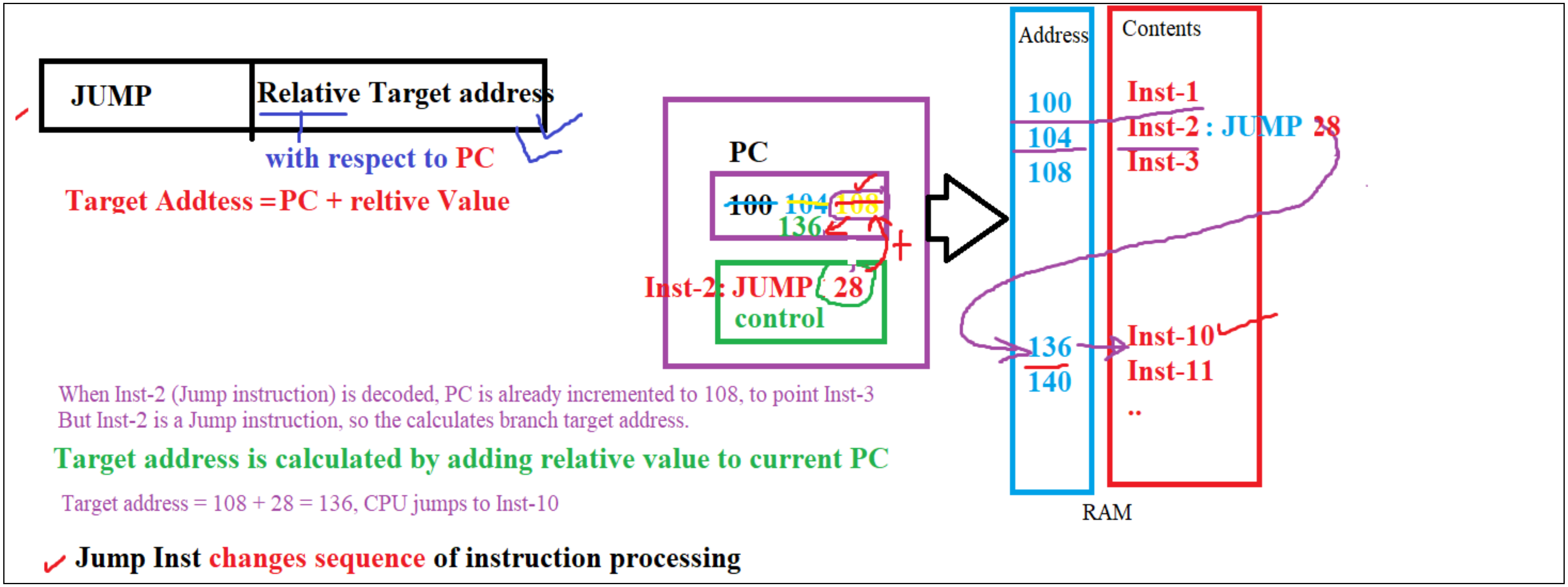
When the JUMP inst is decoded, PC is already incremented to 108 to point Inst-3 but Inst-2 is a JUMP instruction so the PC is loaded with Target address 136. As a result CPU jumps to Inst-10.

Jump Inst changes sequence of instruction processing



Address	Contents
100	Inst-1
104	Inst-2 : JUMP 136
108	Inst-3
136	Inst-10
140	Inst-11
	..

RAM



JUMP	Relative Target address

with respect to PC

Target Addtess =PC + reltive Value

PC

~~100~~ ~~104~~ ~~108~~ 136

+

Inst-2: JUMP 28

control

Address	Contents
100	Inst-1
104	Inst-2 : JUMP 28
108	Inst-3
136	Inst-10
140	Inst-11
	..

RAM

When Inst-2 (Jump instruction) is decoded, PC is already incremented to 108, to point Inst-3
But Inst-2 is a Jump instruction, so the calculates branch target address.

Target address is calculated by adding relative value to current PC

Target address = 108 + 28 = 136, CPU jumps to Inst-10

✓ **Jump Inst changes sequence of instruction processing**

1. Assume a machine using Five-stage RISC pipeline runs a program. **Instruction-3 is a conditional branch instruction**. If the condition is TRUE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is TURE, program control returns to Instruction-4. Show the time steps of pipelining stages assuming that both Instructions 3 and 8 are evaluated FALSE.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Inst-1	IF	ID	ALU	MA	WR																			
Inst-2		IF	ID	ALU	MA	WR																		
Inst-3			IF	ID	ALU	MA	WR																	
Inst-4				IF	ID	ALU	MA	WR																
Inst-5					IF	ID	ALU	MA	WR															
Inst-6						IF	ID	ALU	MA	WR														
Inst-7							IF	ID	ALU	MA	WR													
Inst-8								IF	ID	ALU	MA	WR												
Inst-9									IF	ID	ALU	MA												
Inst-10										IF	ID	ALU												
Inst-11											IF	ID												
Inst-12												IF												
Inst-13																								

2. Assume a machine using Five-stage RISC pipeline runs a program. Instruction-3 is a conditional branch instruction. If the condition is TRUE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is TURE, program control returns to Instruction-4. Show the time steps of pipelining stages assuming that Instruction-3 is evaluated TRUE and Instruction-8 is evaluated FALSE.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Inst-1	IF	ID	ALU	MA	WR																			
Inst-2		IF	ID	ALU	MA	WR																		
Inst-3			IF	ID	ALU	MA	WR																	
Inst-4				IF	ID	ALU																		
Inst-5					IF	ID																		
Inst-6						IF																		
Inst-7							IF	ID	ALU	MA	WR													
Inst-8								IF	ID	ALU	MA	WR												
Inst-9									IF	ID	ALU	MA												
Inst-10										IF	ID	ALU												
Inst-11											IF	ID												
Inst-12												IF												
Inst-13																								

3. Assume a machine using Five-stage CISC pipelining runs a program. Instruction-3 is a conditional branch instruction. If the condition is TRUE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is TURE, program control returns to Instruction-4. Show the time steps of pipelining stages assuming that Instruction-3 is evaluated FALSE and Instruction-8 is evaluated TRUE.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Inst-1	IF	ID	MA	EX	WR																			
Inst-2		IF	ID	MA	EX	WR																		
Inst-3			IF	ID	MA	EX	WR																	
Inst-4				IF	ID	MA	EX	WR				IF	ID	MA	EX	WR				IF	ID	MA	EX	WR
Inst-5					IF	ID	MA	EX	WR				IF	ID	MA	EX	WR				IF	ID	MA	EX
Inst-6						IF	ID	MA	EX	WR				IF	ID	MA	EX	WR				IF	ID	MA
Inst-7							IF	ID	MA	EX	WR				IF	ID	MA	EX	WR				IF	ID
Inst-8								IF	ID	MA	EX	WR				IF	ID	MA	EX	WR				IF
Inst-9									IF	ID	MA						IF	ID	MA					
Inst-10										IF	ID							IF	ID					
Inst-11											IF								IF					
Inst-12																								
Inst-13																								

1. Assume a machine using Five-stage pipelining (IF – ID – MU– EX – WR) runs a program. Instruction-3 is a conditional branch instruction. If the condition is TRUE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is TRUE, program control returns to Instruction-4. Show at least 15-time steps of pipelining stages assuming that both Instructions 3 and 8 are evaluated TRUE.

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Inst-1	IF	ID	MU	EX	WR															
Inst-2		IF	ID	MU	EX	WR														
Inst-3			IF	ID	MU	EX	WR													
Inst-4				IF	ID	MU	CELAR					IF	ID	MU	EX	WR				
Inst-5					IF	ID							IF	ID	MU	EX				
Inst-6						IF								IF	ID	MU				
Inst-7							IF	ID	MU	EX	WR				IF	ID				
Inst-8								IF	ID	MU	EX	WR				IF				
Inst-9									IF	ID	MU	CLEAR								
Inst-10										IF	ID									
Inst-11											IF									
Inst-12																				
Inst-13																				

2. Assume a machine using Five-stage pipelining (IF – ID – MU– EX – WR) runs a program. Instruction-3 is a conditional loop instruction. If the condition is TRUE, CPU runs instructions: 8 - 10 twice. Show the time steps of pipelining stages assuming that the condition is evaluated TRUE. Just comment if the condition is FALSE

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Inst-1	IF	ID	MU	EX	WR															
Inst-2		IF	ID	MU	EX	WR														
Inst-3			IF	ID	MU	EX	WR													
Inst-4				IF	ID	MU	CLEAR													
Inst-5					IF	ID														
Inst-6						IF														
Inst-7																				
Inst-8							IF	ID	MU	EX	WR		IF	ID	MU	EX	WR			
Inst-9								IF	ID	MU	EX	WR		IF	ID	MU	EX	WR		
Inst-10									IF	ID	MU	EX	WR		IF	ID	MU	EX	WR	
Inst-11										IF	ID	MU	CLEAR			IF	ID	MU		
Inst-12											IF	ID					IF	ID		
Inst-13												IF						IF		

1. Assume a machine using Five-stage pipelining (IF – ID – MU– EX – WR) runs a program. Instruction-3 is a conditional branch instruction. If the condition is TRUE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is FALSE, program control returns to Instruction-4. Show at least 15-time steps of pipelining stages assuming that both Instructions 3 and 8 are evaluated TRUE.

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Inst-1	IF	ID	MU	EX	WR															
Inst-2		IF	ID	MU	EX	WR														
Inst-3			IF	ID	MU	EX	WR													
Inst-4				IF	ID	MU	CLEAR													
Inst-5					IF	ID														
Inst-6						IF														
Inst-7							IF	ID	MU	EX	WR									
Inst-8								IF	ID	MU	EX	WR								
Inst-9									IF	ID	MU	EX	WR							
Inst-10										IF	ID	MU	EX	WR						
Inst-11											IF	ID	MU	EX	WR					
Inst-12												IF	ID	MU	EX	WR				
Inst-13													IF	ID	MU	EX				

2. Assume a machine using Five-stage pipelining (IF – ID – MU– EX – WR) runs a program. Instruction-3 is a conditional branch instruction. If the condition is TRUE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is TRUE, program control returns to Instruction-4. Show at least 15-time steps of pipelining stages assuming that both Instructions 3 and 8 are evaluated TRUE.

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Inst-1	IF	ID	MU	EX	WR															
Inst-2		IF	ID	MU	EX	WR														
Inst-3			IF	ID	MU	EX	WR													
Inst-4				IF	ID	MU	CELAR					IF	ID	MU	EX	WR				
Inst-5					IF	ID							IF	ID	MU	EX				
Inst-6						IF								IF	ID	MU				
Inst-7							IF	ID	MU	EX	WR				IF	ID				
Inst-8								IF	ID	MU	EX	WR				IF				
Inst-9									IF	ID	MU	CLEAR								
Inst-10										IF	ID									
Inst-11											IF									
Inst-12																				
Inst-13																				

3. Assume a machine using Five-stage pipelining (IF – ID – MU– EX – WR) runs a program. Instruction-3 is a conditional branch instruction. If the condition is TRUE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is FALSE, program control returns to Instruction-4. Show at least 15-time steps of pipelining stages assuming that Instruction 3 is evaluated FALSE and Instruction 8 is evaluated TRUE.

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Inst-1	IF	ID	MU	EX	WR															
Inst-2		IF	ID	MU	EX	WR														
Inst-3			IF	ID	MU	EX	WR													
Inst-4				IF	ID	MU	EX	WR												
Inst-5					IF	ID	MU	EX	WR											
Inst-6						IF	ID	MU	EX	WR										
Inst-7							IF	ID	MU	EX	WR									
Inst-8								IF	ID	MU	EX	WR								
Inst-9									IF	ID	MU	EX	WR							
Inst-10										IF	ID	MU	EX	WR						
Inst-11											IF	ID	MU	EX	WR					
Inst-12												IF	ID	MU	EX	WR				
Inst-13													IF	ID	MU	EX				

4. Assume a machine using Five-stage pipelining (IF – ID – MU– EX – WR) runs a program. Instruction-3 is a conditional branch instruction. If the condition is TRUE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is FALSE, program control returns to Instruction-4. Show at least 15-time steps of pipelining stages assuming that both Instruction 3 is evaluated FALSE and Instruction 8 is evaluated FALSE.

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Inst-1	IF	ID	MU	EX	WR															
Inst-2		IF	ID	MU	EX	WR														
Inst-3			IF	ID	MU	EX	WR													
Inst-4				IF	ID	MU	EX	WR				IF	ID	MU	EX					
Inst-5					IF	ID	MU	EX	WR				IF	ID	MU	EX				
Inst-6						IF	ID	MU	EX	WR				IF	ID	MU				
Inst-7							IF	ID	MU	EX	WR				IF	ID				
Inst-8								IF	ID	MU	EX	WR				IF				
Inst-9									IF	ID	MU	CLEAR								
Inst-10										IF	ID									
Inst-11											IF									
Inst-12																				
Inst-13																				

5. Assume a machine using Five-stage pipelining (IF – ID – MU– EX – WR) runs a program. Instruction-3 is a conditional branch instruction. If the condition is FALSE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is TRUE, program control returns to Instruction-4. Show at least 15-time steps of pipelining stages assuming that both Instructions 3 and 8 are evaluated TRUE.

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Inst-1	IF	ID	MU	EX	WR															
Inst-2		IF	ID	MU	EX	WR														
Inst-3			IF	ID	MU	EX	WR													
Inst-4				IF	ID	MU	EX	WR				IF	ID	MU	EX					
Inst-5					IF	ID	MU	EX	WR				IF	ID	MU	EX				
Inst-6						IF	ID	MU	EX	WR				IF	ID	MU				
Inst-7							IF	ID	MU	EX	WR				IF	ID				
Inst-8								IF	ID	MU	EX	WR				IF				
Inst-9									IF	ID	MU	CLEAR								
Inst-10										IF	ID									
Inst-11											IF									
Inst-12																				
Inst-13																				

6. Assume a machine using Five-stage pipelining (IF – ID – MU– EX – WR) runs a program. Instruction-3 is a conditional branch instruction. If the condition is FALSE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is TRUE, program control returns to Instruction-4. Show at least 15-time steps of pipelining stages assuming that both Instructions 3 and 8 are evaluated FALSE.

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Inst-1	IF	ID	MU	EX	WR															
Inst-2		IF	ID	MU	EX	WR														
Inst-3			IF	ID	MU	EX	WR													
Inst-4				IF	ID	MU	CLEAR													
Inst-5					IF	ID														
Inst-6						IF														
Inst-7							IF	ID	MU	EX	WR									
Inst-8								IF	ID	MU	EX	WR								
Inst-9									IF	ID	MU	EX	WR							
Inst-10										IF	ID	MU	EX	WR						
Inst-11											IF	ID	MU	EX	WR					
Inst-12												IF	ID	MU	EX	WR				
Inst-13													IF	ID	MU	EX				

7. Assume a machine using Five-stage pipelining (IF – ID – MU– EX – WR) runs a program. Instruction-3 is a conditional loop instruction. If the condition is TRUE, CPU runs instructions: 8 - 10 twice. Show the time steps of pipelining stages assuming that the condition is evaluated TRUE. Just comment if the condition is FALSE

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Inst-1	IF	ID	MU	EX	WR															
Inst-2		IF	ID	MU	EX	WR														
Inst-3			IF	ID	MU	EX	WR													
Inst-4				IF	ID	MU	CLEAR													
Inst-5					IF	ID														
Inst-6						IF														
Inst-7																				
Inst-8							IF	ID	MU	EX	WR		IF	ID	MU	EX	WR			
Inst-9								IF	ID	MU	EX	WR		IF	ID	MU	EX	WR		
Inst-10									IF	ID	MU	EX	WR		IF	ID	MU	EX	WR	
Inst-11										IF	ID	MU	CLEAR			IF	ID	MU		
Inst-12											IF	ID					IF	ID		
Inst-13												IF						IF		

8.	Identify data hazards, if any in the following instruction while processing through a pipelined processor	Just comment how these instructions can be processed in a a pipelined processor.
a)	ADD R1, R2, R3 SUB R4, R1, R5 AND R6, R1, R7 OR R8, R1, R9 XOR R10, R1, R11	
b)	ADD R1, R2, R3 SUB R4, R1, R6	
c)	SUB R4, R1, R3 ADD R1, R2, R3 MUL R6, R1, R7	

For the following code, identify data hazard, structural hazard and control hazard, if any in consecutive instructions.

```
add r3, r1, r2 ;r3 destination
and r5, r3, r4 ;r5 result field
load r6, 24(r3) ;r6 destination
add r2, r6, r3 ;r2 result field
store r6, 12(r2) ;r6 source
```

For the following code, identify data hazard, structural hazard and control hazard, if any in consecutive instructions.

```
load r6, 24(r3);r6 destination
add r3, r1, r6 ;r3 result field
And r5, r3, r4 ;r5 result field
add r2, r6, r5 ;r2 result field
store r6, 12(r2);
```

For the code sequence below, state whether it must stall, can avoid stalls using only forwarding, or can execute without stalling or forwarding.

```
add R1,R0,1
add R2,R0,2
add R3,R0,2
add R3,R0,4
add R5,R0,5
```

No stall