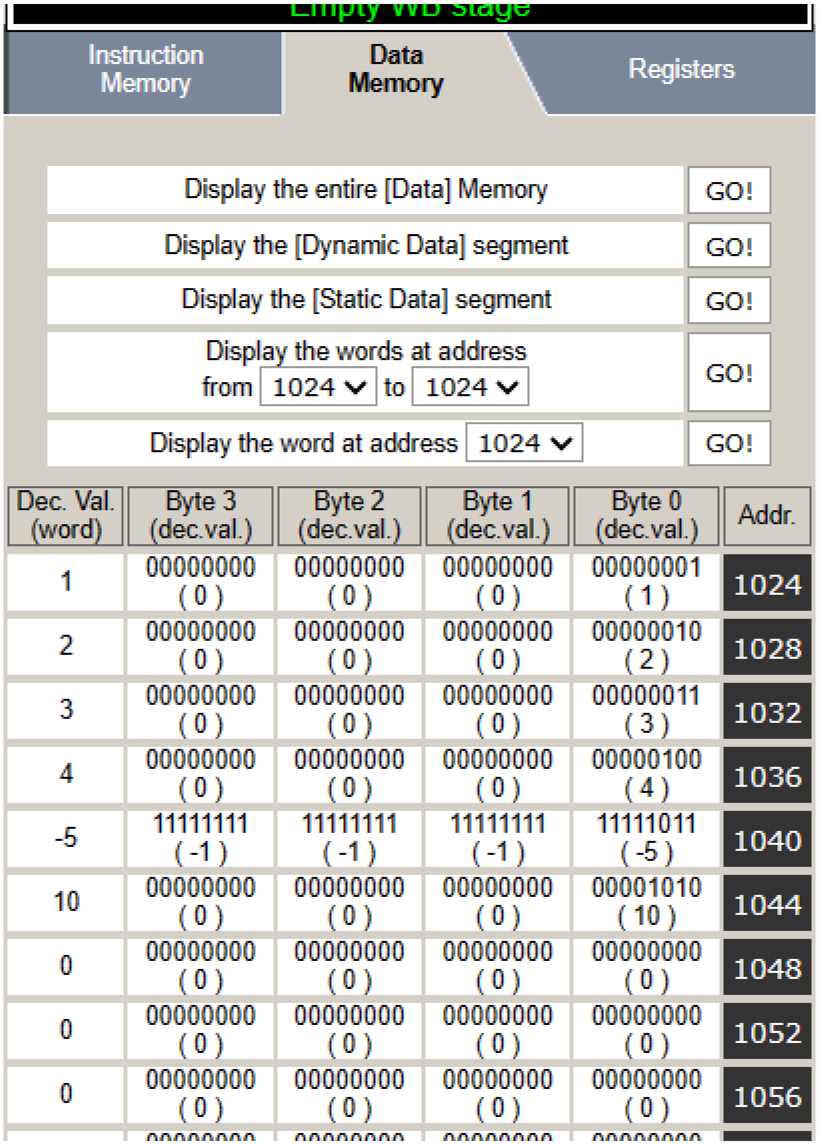
Question 1

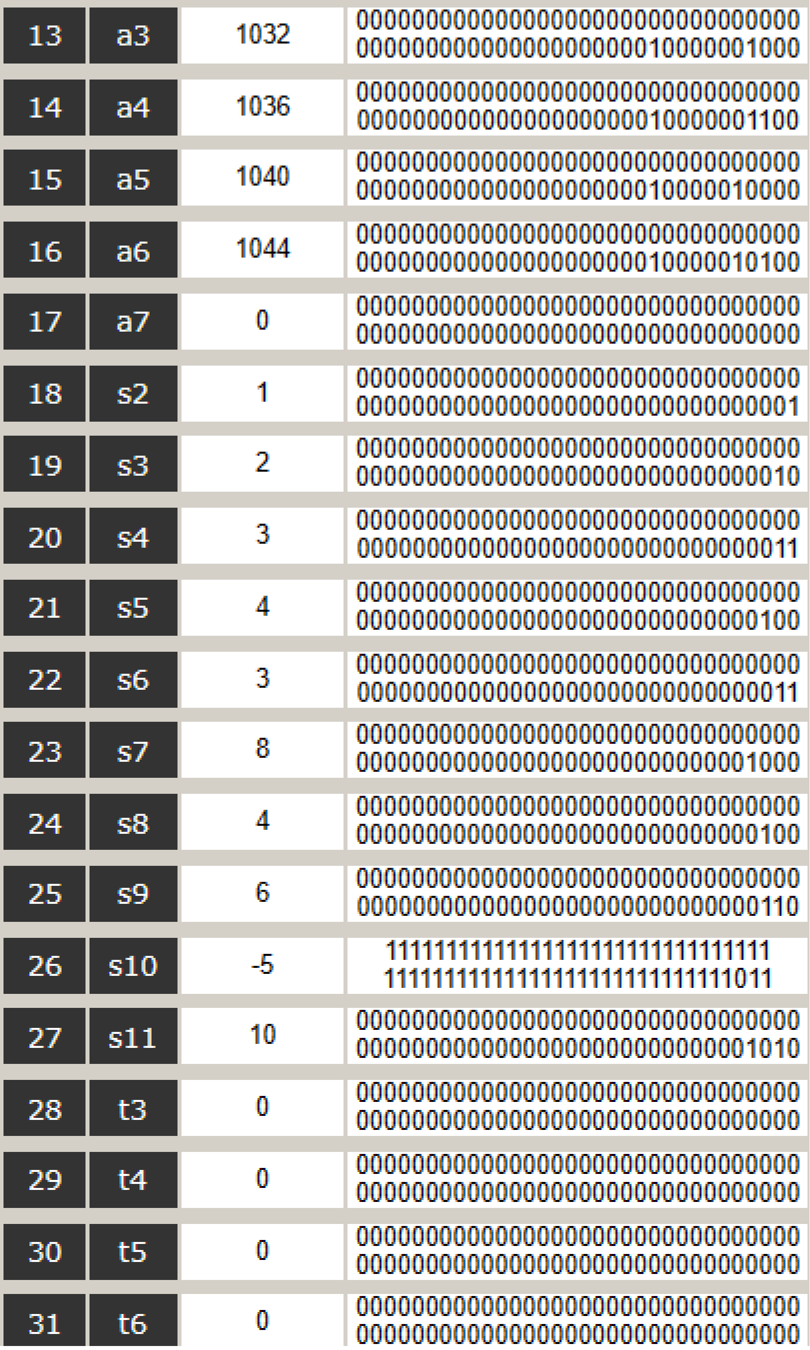
Input a+ib=1+2i, c+id=3+4i

The inputs and outputs stored in register and the data memory for all the operations


Empty WB stage
Instruction Memory
Data Memory
Registers
Display the entire [Data] Memory
GO!
Display the [Dynamic Data] segment
GO!
Display the [Static Data] segment
GO!
Display the words at address from 1024 to 1024
GO!
Display the word at address 1024
GO!

Dec. Val. (word)	Byte 3 (dec.val.)	Byte 2 (dec.val.)	Byte 1 (dec.val.)	Byte 0 (dec.val.)	Addr.
1	00000000 (0)	00000000 (0)	00000000 (0)	00000001 (1)	1024
2	00000000 (0)	00000000 (0)	00000000 (0)	00000010 (2)	1028
3	00000000 (0)	00000000 (0)	00000000 (0)	00000011 (3)	1032
4	00000000 (0)	00000000 (0)	00000000 (0)	00000100 (4)	1036
-5	11111111 (-1)	11111111 (-1)	11111111 (-1)	11111011 (-5)	1040
10	00000000 (0)	00000000 (0)	00000000 (0)	00001010 (10)	1044
0	00000000 (0)	00000000 (0)	00000000 (0)	00000000 (0)	1048
0	00000000 (0)	00000000 (0)	00000000 (0)	00000000 (0)	1052
0	00000000 (0)	00000000 (0)	00000000 (0)	00000000 (0)	1056

Stored in data memory



13	a3	1032	00000000000000000000000000000000 00000000000000000000010000001000
14	a4	1036	00000000000000000000000000000000 00000000000000000000010000001100
15	a5	1040	00000000000000000000000000000000 00000000000000000000010000010000
16	a6	1044	00000000000000000000000000000000 00000000000000000000010000010100
17	a7	0	00000000000000000000000000000000 00000000000000000000000000000000
18	s2	1	00000000000000000000000000000000 00000000000000000000000000000001
19	s3	2	00000000000000000000000000000000 00000000000000000000000000000010
20	s4	3	00000000000000000000000000000000 00000000000000000000000000000011
21	s5	4	00000000000000000000000000000000 00000000000000000000000000000100
22	s6	3	00000000000000000000000000000000 00000000000000000000000000000011
23	s7	8	00000000000000000000000000000000 00000000000000000000000000001000
24	s8	4	00000000000000000000000000000000 00000000000000000000000000000100
25	s9	6	00000000000000000000000000000000 00000000000000000000000000000110
26	s10	-5	11111111111111111111111111111111 11111111111111111111111111111011
27	s11	10	00000000000000000000000000000000 00000000000000000000000000001010
28	t3	0	00000000000000000000000000000000 00000000000000000000000000000000
29	t4	0	00000000000000000000000000000000 00000000000000000000000000000000
30	t5	0	00000000000000000000000000000000 00000000000000000000000000000000
31	t6	0	00000000000000000000000000000000 00000000000000000000000000000000

Stored in Registers

The Execution tables

All the execution table's link are in here **Execution Table**

Code for Flushing with or without Forwarding

```
#data
.data
no1_real: .word 1
no1_img:  .word 2
no2_real: .word 3
no2_img:  .word 4
ans_real: .word 0
ans_img:  .word 0


.text
#Loading the addresses in register
la a1,no1_real
la a2,no1_img
la a3,no2_real
la a4,no2_img
la a5, ans_real
la a6, ans_img

#loading the values stored in registers to the register
lw s2, 0(a1) # 1st number real part
lw s3, 0(a2) # 1st number complex part
lw s4, 0(a3) # 2nd number real part
lw s5, 0(a4) # 2nd number complex part

# Perform complex multiplication: (a+bi) * (c+di) = (ac - bd) + (ad + bc)i

mv t3,s2
mv t4,s3
mv t5,s4
mv t6,s5
```

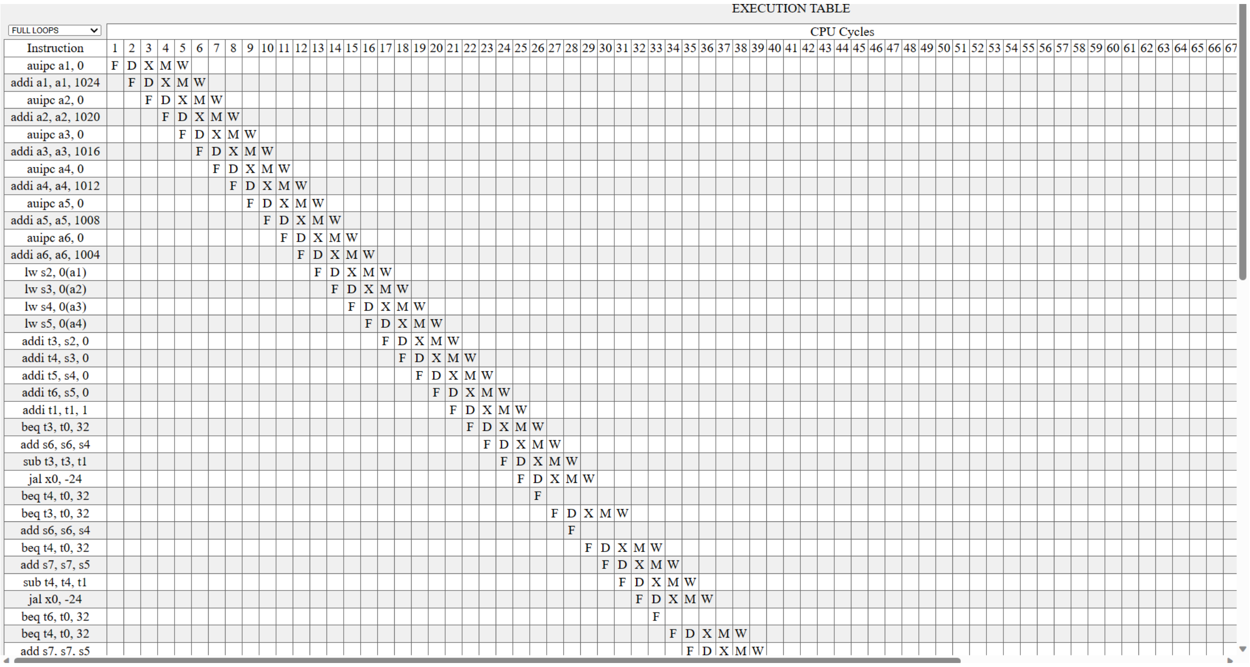
```
li t0,0
li t1,1
# Calculate (ac) and (bd)

loop1:
beq t3,t0, ans1
add s6,s6,s4
sub t3,t3,t1
j loop1
ans1:
loop2:
beq t4, t0,ans2
add s7,s7,s5
sub t4,t4,t1
j loop2
ans2:
loop3:
# Calculate (ad) and (bc)
beq t6,t0,ans3
add s8,s8,s2
sub t6,t6,t1
j loop3
ans3:
loop4:
beq t5,t0, ans4
add s9,s9,s3
sub t5,t5,t1
j loop4

ans4:
# Calculate real part of the result: (ac - bd)
sub s10,s6,s7
# Calculate real part of the result: (ad+bc)
add s11,s8,s9
```

```
#storing the values to data memory addresses
sw s10,0(a5)
sw s11,0(a6)
```

Forward Activated Flushing is there

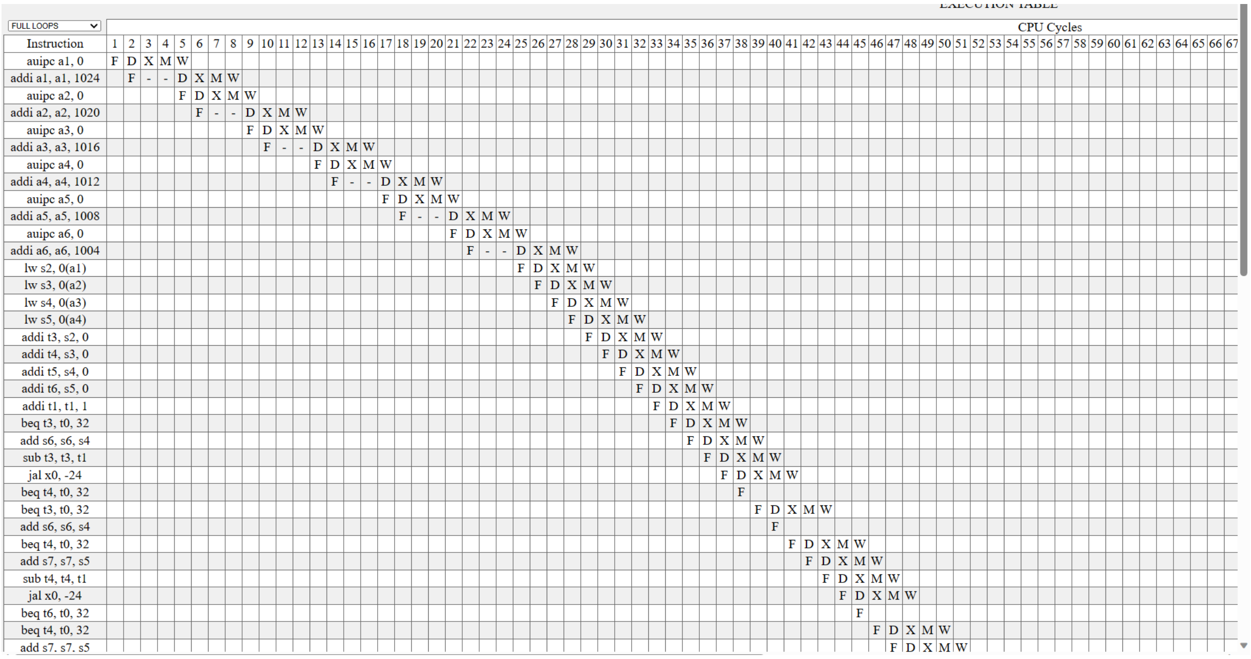


EXECUTION TABLE

FULL LOOPS	CPU Cycles																																																																
Instruction	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	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67
auipc a1, 0	F	D	X	M	W																																																														
addi a1, a1, 1024		F	D	X	M	W																																																												
auipc a2, 0			F	D	X	M	W																																																										
addi a2, a2, 1020				F	D	X	M	W																																																								
auipc a3, 0					F	D	X	M	W																																																							
addi a3, a3, 1016						F	D	X	M	W																																																						
auipc a4, 0							F	D	X	M	W																																																				
addi a4, a4, 1012								F	D	X	M	W																																																		
auipc a5, 0									F	D	X	M	W																																																	
addi a5, a5, 1008										F	D	X	M	W																																																
auipc a6, 0											F	D	X	M	W																																															
addi a6, a6, 1004												F	D	X	M	W																																														
lw s2, 0(a1)													F	D	X	M	W																																													
lw s3, 0(a2)														F	D	X	M	W																																												
lw s4, 0(a3)															F	D	X	M	W																																											
lw s5, 0(a4)																F	D	X	M	W																																										
addi t3, s2, 0																	F	D	X	M	W																																										
addi t4, s3, 0																		F	D	X	M	W																																									
addi t5, s4, 0																			F	D	X	M	W																																								
addi t6, s5, 0																				F	D	X	M	W																																							
addi t1, t1, 1																					F	D	X	M	W																																						
beq t3, t0, 32																						F	D	X	M	W																																					
add s6, s6, s4																							F	D	X	M	W																																				
sub t3, t3, t1																								F	D	X	M	W																																			
jal x0, -24																									F	D	X	M	W																																		
beq t4, t0, 32																										F																																										
beq t3, t0, 32																											F	D	X	M	W																																
add s6, s6, s4																												F																																						
beq t4, t0, 32																													F	D	X	M	W																														
add s7, s7, s5																														F	D	X	M	W																													
sub t4, t4, t1																															F	D	X	M	W																												
jal x0, -24																																F	D	X	M	W																											
beq t6, t0, 32																																	F																																		
beq t4, t0, 32																																		F	D	X	M	W																									
add s7, s7, s5																																			F	D	X	M	W																								

83 Instruction 87 Cycles
Throughput = 83/87=0.954

Forward Deactivated Flushing is there



EXECUTION TABLE

FULL LOOPS

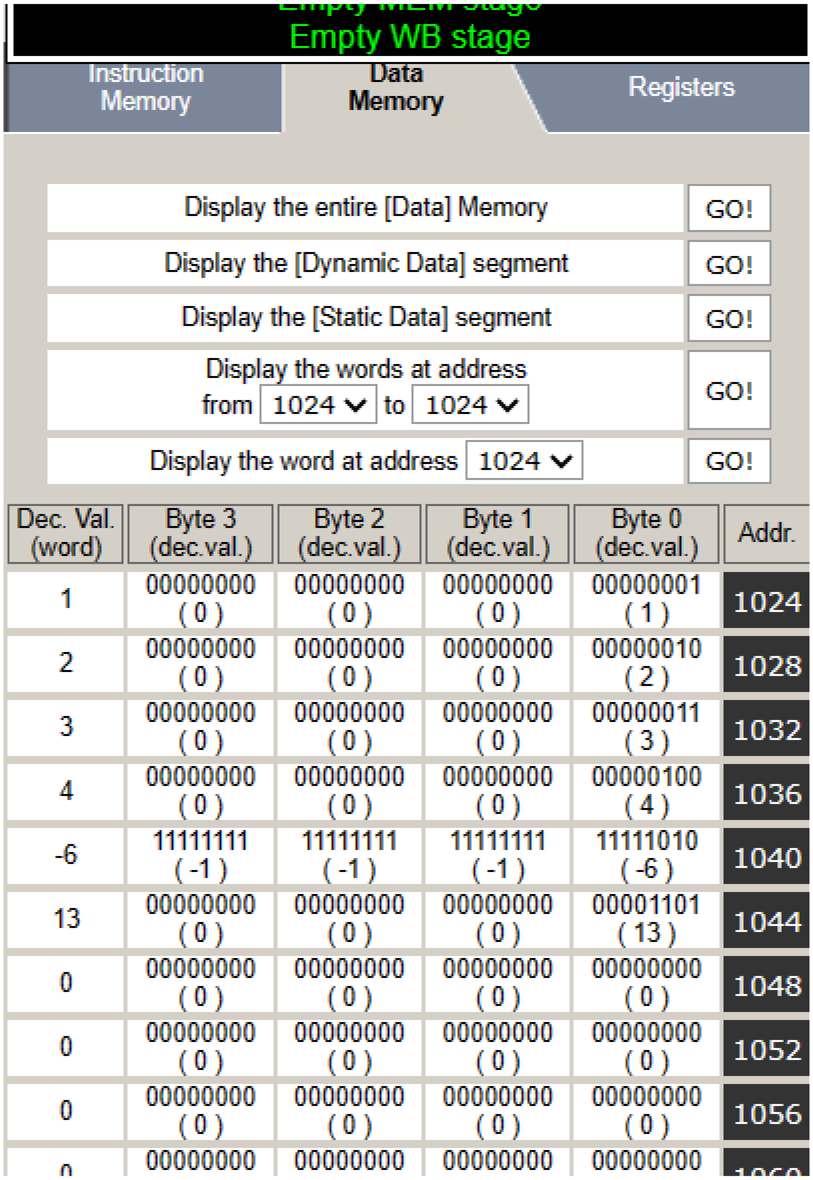
CPU Cycles

Instruction	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	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67
auipc a1, 0	F	D	X	M	W																																																															
addi a1, a1, 1024		F	-	-	D	X	M	W																																																										
auipc a2, 0					F	D	X	M	W																																																									
addi a2, a2, 1020						F	-	-	D	X	M	W																																																						
auipc a3, 0									F	D	X	M	W																																																					
addi a3, a3, 1016										F	-	-	D	X	M	W																																																		
auipc a4, 0													F	D	X	M	W																																																	
addi a4, a4, 1012														F	-	-	D	X	M	W																																														
auipc a5, 0																	F	D	X	M	W																																													
addi a5, a5, 1008																		F	-	-	D	X	M	W																																										
auipc a6, 0																					F	D	X	M	W																																									
addi a6, a6, 1004																						F	-	-	D	X	M	W																																						
lw s2, 0(a1)																									F	D	X	M	W																																					
lw s3, 0(a2)																										F	D	X	M	W																																				
lw s4, 0(a3)																											F	D	X	M	W																																			
lw s5, 0(a4)																												F	D	X	M	W																																		
addi t3, s2, 0																													F	D	X	M	W																																	
addi t4, s3, 0																														F	D	X	M	W																																
addi t5, s4, 0																															F	D	X	M	W																															
addi t6, s5, 0																																F	D	X	M	W																														
addi t1, t1, 1																																	F	D	X	M	W																													
beq t3, t0, 32																																		F	D	X	M	W																												
add s6, s6, s4																																			F	D	X	M	W																											
sub t3, t3, t1																																				F	D	X	M	W																										
jal x0, -24																																					F	D	X	M	W																									
beq t4, t0, 32																																						F																												
beq t3, t0, 32																																							F	D	X	M	W																							
add s6, s6, s4																																								F																										
beq t4, t0, 32																																									F	D	X	M	W																					
add s7, s7, s5																																										F	D	X	M	W																				
sub t4, t4, t1																																											F	D	X	M	W																			
jal x0, -24																																												F	D	X	M	W																		
beq t6, t0, 32																																													F																					
beq t4, t0, 32																																														F	D	X	M	W																
add s7. s7. s5																																															F	D	X	M	W															

83 instructions 108 Cycles

Throughput=83/108=0.768

Without nop in the code as used above for no Flushing I was getting wrong outputs in the registers


Empty MEM stage
Empty WB stage
Instruction Memory
Data Memory
Registers
Display the entire [Data] Memory
GO!
Display the [Dynamic Data] segment
GO!
Display the [Static Data] segment
GO!
Display the words at address
from 1024 to 1024
GO!
Display the word at address 1024
GO!

Dec. Val. (word)	Byte 3 (dec.val.)	Byte 2 (dec.val.)	Byte 1 (dec.val.)	Byte 0 (dec.val.)	Addr.
1	00000000 (0)	00000000 (0)	00000000 (0)	00000001 (1)	1024
2	00000000 (0)	00000000 (0)	00000000 (0)	00000010 (2)	1028
3	00000000 (0)	00000000 (0)	00000000 (0)	00000011 (3)	1032
4	00000000 (0)	00000000 (0)	00000000 (0)	00000100 (4)	1036
-6	11111111 (-1)	11111111 (-1)	11111111 (-1)	11111010 (-6)	1040
13	00000000 (0)	00000000 (0)	00000000 (0)	00001101 (13)	1044
0	00000000 (0)	00000000 (0)	00000000 (0)	00000000 (0)	1048
0	00000000 (0)	00000000 (0)	00000000 (0)	00000000 (0)	1052
0	00000000 (0)	00000000 (0)	00000000 (0)	00000000 (0)	1056
0	00000000	00000000	00000000	00000000	1060

Code For No Flushing with or without forwarding we have to add nop to get proper output in the resistors

```
#data
.data
no1_real: .word 1
no1_img:  .word 2
no2_real: .word 3
no2_img:  .word 4
```

```
ans_real: .word 0
ans_img:  .word 0


.text
#Loading the addresses in register
la a1,no1_real
la a2,no1_img
la a3,no2_real
la a4,no2_img
la a5, ans_real
la a6, ans_img

#loading the values stored in registers to the register
lw s2, 0(a1) # 1st number real part
lw s3, 0(a2) # 1st number complex part
lw s4, 0(a3) # 2nd number real part
lw s5, 0(a4) # 2nd number complex part

# Perform complex multiplication: (a+bi) * (c+di) = (ac - bd) + (ad + bc)i

mv t3,s2
mv t4,s3
mv t5,s4
mv t6,s5

li t0,0
li t1,1
# Calculate (ac) and (bd)

loop1:
beq t3,t0, ans1
nop
add s6,s6,s4
```

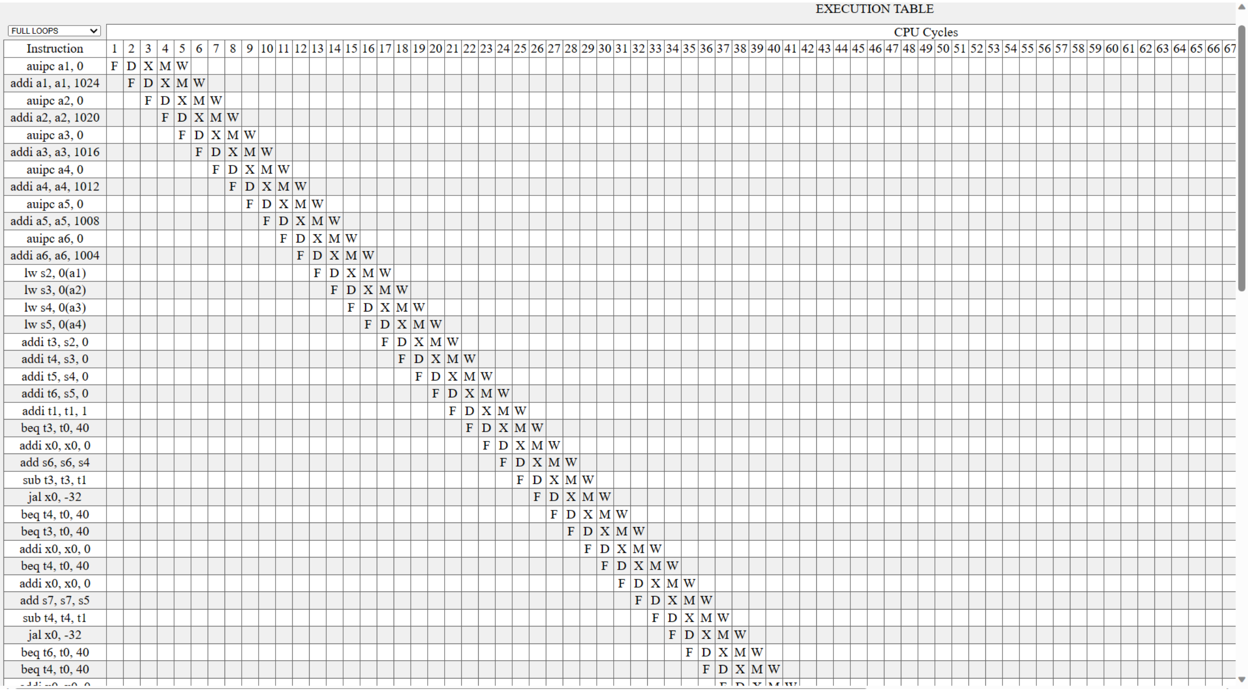
```
sub t3,t3,t1
j loop1
ans1:
loop2:
beq t4, t0,ans2
nop
add s7,s7,s5
sub t4,t4,t1
j loop2
ans2:
loop3:
# Calculate (ad) and (bc)
beq t6,t0,ans3
nop
add s8,s8,s2
sub t6,t6,t1
j loop3
ans3:
loop4:
beq t5,t0, ans4
nop
add s9,s9,s3
sub t5,t5,t1
j loop4

ans4:
# Calculate real part of the result: (ac - bd)
sub s10,s6,s7
# Calculate real part of the result: (ad+bc)
add s11,s8,s9


#storing the values to data memory addresses
sw s10,0(a5)
```

sw s11,0(a6)

Without flushing that is Deactivating flush Forwarding



EXECUTION TABLE

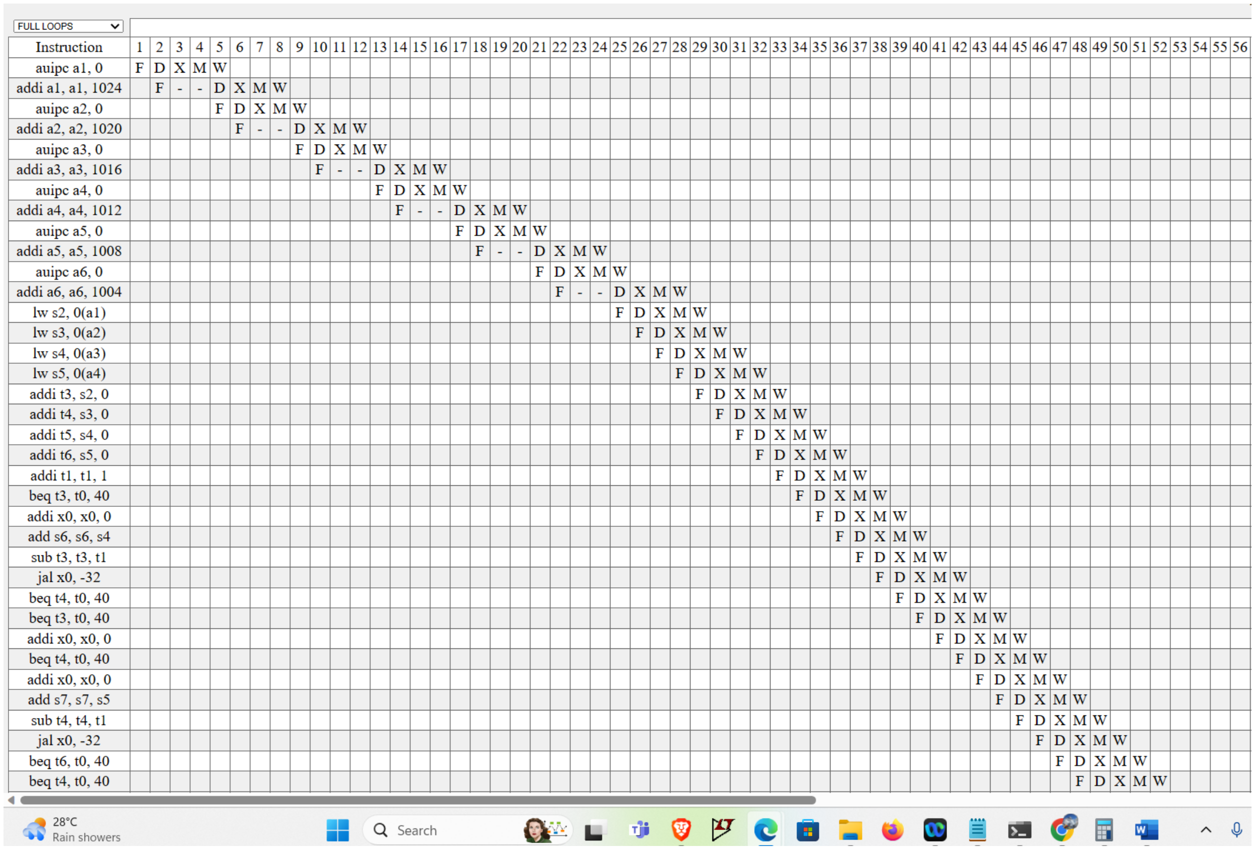
FULL LOOPS

	CPU Cycles																																																																	
Instruction	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	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67
auipc a1, 0	F	D	X	M	W																																																												
addi a1, a1, 1024		F	D	X	M	W																																																											
auipc a2, 0			F	D	X	M	W																																																									
addi a2, a2, 1020				F	D	X	M	W																																																								
auipc a3, 0					F	D	X	M	W																																																							
addi a3, a3, 1016						F	D	X	M	W																																																						
auipc a4, 0							F	D	X	M	W																																																					
addi a4, a4, 1012								F	D	X	M	W																																																				
auipc a5, 0									F	D	X	M	W																																																			
addi a5, a5, 1008										F	D	X	M	W																																																		
auipc a6, 0											F	D	X	M	W																																																	
addi a6, a6, 1004												F	D	X	M	W																																																
lw s2, 0(a1)													F	D	X	M	W																																															
lw s3, 0(a2)														F	D	X	M	W																																														
lw s4, 0(a3)															F	D	X	M	W																																													
lw s5, 0(a4)																F	D	X	M	W																																												
addi t3, s2, 0																	F	D	X	M	W																																											
addi t4, s3, 0																		F	D	X	M	W																																										
addi t5, s4, 0																			F	D	X	M	W																																									
addi t6, s5, 0																				F	D	X	M	W																																								
addi t1, t1, 1																					F	D	X	M	W																																							
beq t3, t0, 40																						F	D	X	M	W																																						
addi x0, x0, 0																							F	D	X	M	W																																				
add s6, s6, s4																								F	D	X	M	W																																			
sub t3, t3, t1																									F	D	X	M	W																																		
jal x0, -32																										F	D	X	M	W																																	
beq t4, t0, 40																											F	D	X	M	W																																
beq t3, t0, 40																												F	D	X	M	W																															
addi x0, x0, 0																													F	D	X	M	W																														
beq t4, t0, 40																														F	D	X	M	W																													
addi x0, x0, 0																															F	D	X	M	W																												
add s7, s7, s5																																F	D	X	M	W																											
sub t4, t4, t1																																	F	D	X	M	W																										
jal x0, -32																																		F	D	X	M	W																									
beq t6, t0, 40																																			F	D	X	M	W																								
beq t4, t0, 40																																				F	D	X	M	W																							

Instruction =93 in 97 Clock Cycles

Throughput =0.9587

Without flushing that is Deactivating flush deactivated forwarding



Instruction	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	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
auipc a1, 0	F	D	X	M	W																																																			
addi a1, a1, 1024		F	-	-	D	X	M	W																																																
auipc a2, 0					F	D	X	M	W																																															
addi a2, a2, 1020						F	-	-	D	X	M	W																																												
auipc a3, 0									F	D	X	M	W																																											
addi a3, a3, 1016										F	-	-	D	X	M	W																																								
auipc a4, 0													F	D	X	M	W																																							
addi a4, a4, 1012														F	-	-	D	X	M	W																																				
auipc a5, 0																	F	D	X	M	W																																			
addi a5, a5, 1008																		F	-	-	D	X	M	W																																
auipc a6, 0																					F	D	X	M	W																															
addi a6, a6, 1004																						F	-	-	D	X	M	W																												
lw s2, 0(a1)																									F	D	X	M	W																											
lw s3, 0(a2)																										F	D	X	M	W																										
lw s4, 0(a3)																											F	D	X	M	W																									
lw s5, 0(a4)																												F	D	X	M	W																								
addi t3, s2, 0																													F	D	X	M	W																							
addi t4, s3, 0																														F	D	X	M	W																						
addi t5, s4, 0																															F	D	X	M	W																					
addi t6, s5, 0																																F	D	X	M	W																				
addi t1, t1, 1																																	F	D	X	M	W																			
beq t3, t0, 40																																		F	D	X	M	W																		
addi x0, x0, 0																																			F	D	X	M	W																	
add s6, s6, s4																																				F	D	X	M	W																
sub t3, t3, t1																																					F	D	X	M	W															
jal x0, -32																																						F	D	X	M	W														
beq t4, t0, 40																																							F	D	X	M	W													
beq t3, t0, 40																																								F	D	X	M	W												
addi x0, x0, 0																																									F	D	X	M	W											
beq t4, t0, 40																																										F	D	X	M	W										
addi x0, x0, 0																																											F	D	X	M	W									
add s7, s7, s5																																												F	D	X	M	W								
sub t4, t4, t1																																													F	D	X	M	W							
jal x0, -32																																														F	D	X	M	W						
beq t6, t0, 40																																															F	D	X	M	W					
beq t4, t0, 40																																																F	D	X	M	W				

Instructions 93 in 118 Clock cycle
Throughput=0.788

To maximize throughput in a processor pipeline, we generally want to avoid pipeline stalls and hazards as much as possible.
1. Forward: Forwarding (also known as data forwarding or data hazard forwarding) is a technique that allows the result of an instruction to be passed directly to another instruction that depends on it, without waiting for the result to be written to a register. This helps in reducing pipeline stalls and improving throughput.

2. Flush: Flushing the pipeline means discarding the instructions in the pipeline due to a mispredicted branch or other control hazard. Flushing can reduce throughput as it leads to wasted work.

We can see that the maximum throughput occurs by enabling forwarding (with or without flushing) as it minimizes pipeline stalls and data hazards and it helps to maintain a higher instruction execution rate and minimize stalls due to dependencies. So, "Forward" would be the choice that contributes the most to maximizing throughput.

CONTROL HAZARD

The control Hazard occurs due to conditional branch

1. Control Hazard at `loop1`:
- The `beq t3, t0, ans1` instruction introduces a control hazard. The decision of whether to branch or not depends on the value of `t3`. The outcome is uncertain until `t3` is evaluated. This can potentially stall the pipeline until the branch is resolved.

2. Control Hazard at `loop2`:
- Similar to `loop1`, the `beq t4, t0, ans2` instruction introduces a control hazard. The decision to branch or not depends on the value of `t4`. The pipeline may stall until the branch is resolved.

3. Control Hazard at `loop3`:
- In this loop, the `beq t6, t0, ans3` instruction introduces a control hazard. The branch outcome depends on the value of `t6`, and the pipeline may stall until the branch is resolved.

4. Control Hazard at `loop4`:
- The `beq t5, t0, ans4` instruction in this loop introduces a control hazard. The decision to branch or not depends on the value of `t5`, and the pipeline may stall until the branch is resolved.

These control hazards can potentially stall the pipeline's execution until the branch conditions are determined, which can impact the overall efficiency of the processor's execution. Techniques like branch prediction are commonly used to mitigate control hazards and reduce pipeline stalls.

DATA HAZARD

In the program, there are several data hazards of the "Read-After-Write" (RAW) type. These hazards occur when an instruction depends on the result of a previous instruction that hasn't been written back to the register file yet. Here are the locations of data hazards and their types:

1. RAW Data Hazard (Read-After-Write):

- After each `lw` instruction (loading values from memory), there is a RAW hazard with the subsequent `mv` instruction.

- For example, after the `lw s2, 0(a1)` instruction (loading the first number's real part), there is a RAW hazard between the `mv t3, s2` instruction because `mv` reads the value from `s2` which is not yet available.

- Similarly, after the `lw s3, 0(a2)` instruction, there is a RAW hazard between the `mv t4, s3` instruction.

- The `lw` instructions for the second complex number, `lw s4, 0(a3)` and `lw s5, 0(a4)`, also introduce RAW hazards with the subsequent `mv` instructions.

- In addition, there are RAW hazards between the `mv` instructions and the `beq` instructions (`t3`, `t4`, `t6`, and `t5` are used in the condition).

Question 2

Based on the provided output and the given branch prediction strategies, here's how each strategy affected the overall performance of the program:

1. Always Taken (AT):

```
The content of A is:
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9
The content of B is:
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
The content of C=A*B is:
0 0 0 0 0 0 0 0 0 0
```

```
0 10 20 30 40 50 60 70 80 90
0 20 40 60 80 100 120 140 160 180
0 30 60 90 120 150 180 210 240 270
0 40 80 120 160 200 240 280 320 360
0 50 100 150 200 250 300 350 400 450
0 60 120 180 240 300 360 420 480 540
0 70 140 210 280 350 420 490 560 630
0 80 160 240 320 400 480 560 640 720
0 90 180 270 360 450 540 630 720 810
```

------------ STATISTICS -----------

- **Number of Instructions: 225452**
- **Number of Cycles: 324562**
- **Branch Prediction Accuracy: 62.35%**
- **Number of Cycles: 324562**
- **Average Cycles per Instruction: 1.4396**
- **Control Hazards: 29662**
- **Data Hazards: 105128**
- **Memory Hazards: 11735**

2. Always Not Taken (NT):

```
The content of A is:
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9
The content of B is:
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
```

```
The content of C=A*B is:
0 0 0 0 0 0 0 0 0 0
0 10 20 30 40 50 60 70 80 90
0 20 40 60 80 100 120 140 160 180
0 30 60 90 120 150 180 210 240 270
0 40 80 120 160 200 240 280 320 360
0 50 100 150 200 250 300 350 400 450
0 60 120 180 240 300 360 420 480 540
0 70 140 210 280 350 420 490 560 630
0 80 160 240 320 400 480 560 640 720
0 90 180 270 360 450 540 630 720 810
Program exit from an exit() system call
     ------------ STATISTICS -----------
```

- **Number of Instructions: 225440**
- **Number of Cycles: 318756**
- **Avg Cycles per Instrcution: 1.4139**
- **Branch Prediction Accuracy: 37.65%**
- **Number of Cycles: 318756**
- **Average Cycles per Instruction: 1.4139**
- **Control Hazards: 40678**
- **Data Hazards: 110957**
- **Memory Hazards: 11735**

3. Back Taken Forward Not Taken (BTFNT):

```
he content of A is:
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9
The content of B is:
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
```

```
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
The content of C=A*B is:
0 0 0 0 0 0 0 0 0 0
0 10 20 30 40 50 60 70 80 90
0 20 40 60 80 100 120 140 160 180
0 30 60 90 120 150 180 210 240 270
0 40 80 120 160 200 240 280 320 360
0 50 100 150 200 250 300 350 400 450
0 60 120 180 240 300 360 420 480 540
0 70 140 210 280 350 420 490 560 630
0 80 160 240 320 400 480 560 640 720
0 90 180 270 360 450 540 630 720 810
Program exit from an exit() system call
```

------------ STATISTICS -----------

- **Number of Instructions: 225452**
- **Number of Cycles: 313226**
- **Avg Cycles per Instrcution: 1.3893**
- **Branch Prediction Accuracy: 63.25%**
- **Number of Cycles: 313226**
- **Average Cycles per Instruction: 1.3893**
- **Control Hazards: 29258**
- **Data Hazards: 110841**
- **Memory Hazards: 11735**

4. Branch Prediction Buffer (BPB):

```
The content of A is:
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9
The content of B is:
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
```

```
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
The content of C=A*B is:
0 0 0 0 0 0 0 0 0 0
0 10 20 30 40 50 60 70 80 90
0 20 40 60 80 100 120 140 160 180
0 30 60 90 120 150 180 210 240 270
0 40 80 120 160 200 240 280 320 360
0 50 100 150 200 250 300 350 400 450
0 60 120 180 240 300 360 420 480 540
0 70 140 210 280 350 420 490 560 630
0 80 160 240 320 400 480 560 640 720
0 90 180 270 360 450 540 630 720 810
Program exit from an exit() system call
        ------------ STATISTICS -----------
```

- **Number of Instructions: 225452**
- **Number of Cycles: 318552**
- **Avg Cycles per Instrcution: 1.4129**
- **Branch Prediction Accuracy: 62.75%**
- **Number of Cycles: 318552**
- **Average Cycles per Instruction: 1.4129**
- **Control Hazards: 29482**
- **Data Hazards: 108238**
- **Memory Hazards: 11735**

Explanation:

- Always Taken (AT): This strategy assumes all branches are taken. While it had a decent prediction accuracy, it still incurred a significant number of control hazards.

- Always Not Taken (NT): This strategy assumes no branches are taken. The low prediction accuracy led to a high number of control hazards, making it less efficient than the Always Taken strategy.

- Back Taken Forward Not Taken (BTFNT): This strategy uses a combination of taken and not taken predictions based on the previous branch outcomes. It provided a relatively better prediction accuracy, reducing the number of control hazards compared to the Always Taken and Always Not Taken strategies.

- Branch Prediction Buffer (BPB): This strategy employs a branch prediction buffer with 2-bit history information. It offered a good balance between prediction accuracy and control hazards, making it more efficient in terms of the overall program execution.

In summary, the Branch Prediction Buffer strategy showed the best overall performance among the four strategies, providing a reasonable prediction accuracy while minimizing the number of control hazards.