

# **LAPORAN PRAKTIKUM ARSITEKTUR DAN ORGANISASI KOMPUTER**

## **CPU SIMULATOR**



**Agus Pranata Marpaung  
13323033  
DIII Teknologi Komputer**

**INSTITUT TEKNOLOGI DEL  
FAKULTAS VOKASI**

## Judul Praktikum

---

<b>Minggu/Sesi</b>	:	V/2
<b>Kode Mata Kuliah</b>	:	1031103
<b>Nama Mata Kuliah</b>	:	ARSITEKTUR DAN ORGANISASI KOMPUTER
<b>Setoran</b>	:	Laporan Materi CPU Simulator dikirimkan dalam bentuk PDF dan beri nama “ <i>Prak4_CPU_Simulator_NIM.pdf</i> ”
<b>Batas Waktu Setoran</b>	:	<i>3 Oktober 2023 jam 8:00</i>
<b>Tujuan</b>	:	<ol style="list-style-type: none"><li>1. <i>Use the CPU simulator to create basic CPU instructions.</i></li><li>2. <i>Use the simulator to execute basic CPU instructions.</i></li><li>3. <i>Use CPU instructions to move data to registers, compare values in registers, push data to the stack, pop data from the stack, jump to address locations and add values held in registers.</i></li><li>4. <i>Explain the functions of special CPU registers such as the PC, SR and SP registers.</i></li></ol>

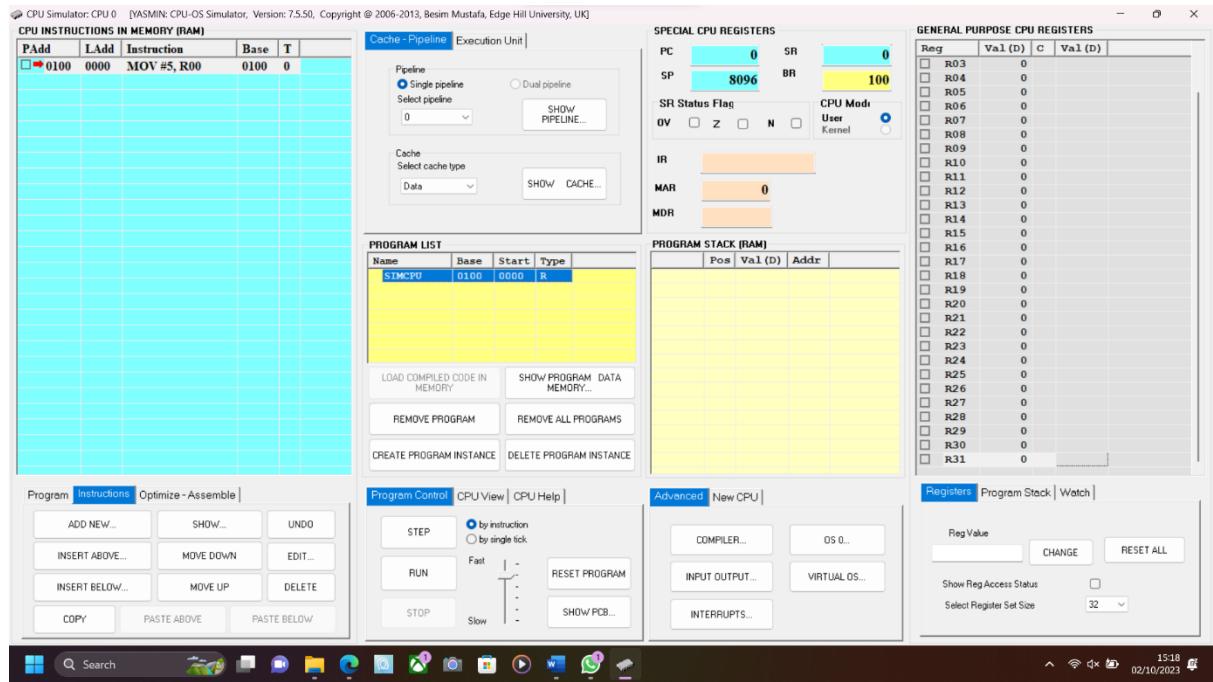
### Petunjuk

1. Laporan praktikum dikerjakan secara individu (tidak berkelompok)
2. Setiap individu diperbolehkan memberikan pertanyaan dan diskusi melalui WAG pada sesi kedua di hari praktikum
3. Laporan praktikum akan dikirimkan sesi praktikum melalui e-course dan mengikutiformat yang telah disediakan sebelumnya
4. Tidak ada toleransi keterlambatan, jika terlambat maka akan terjadi perngurangan nilai.
5. Dalam pengerajan laporan praktikum, dilarang keras melakukan plagiasi(mencontek).

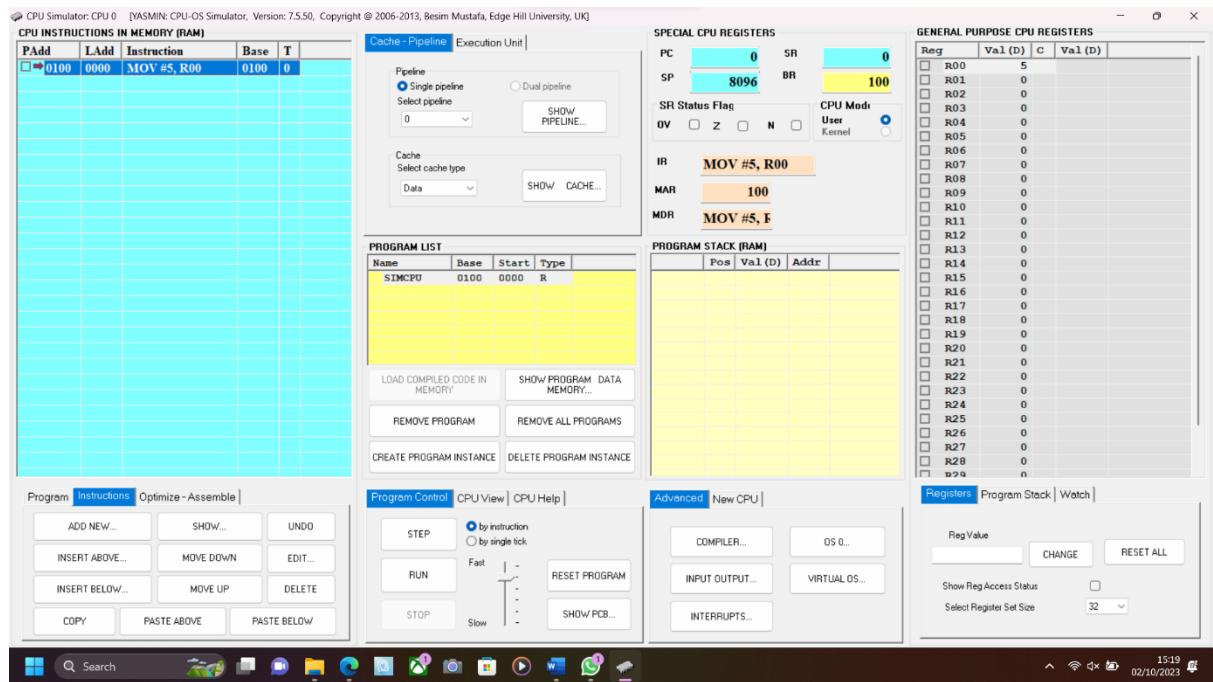
## Arsitektur dan Organisasi Komputer

### Tugas

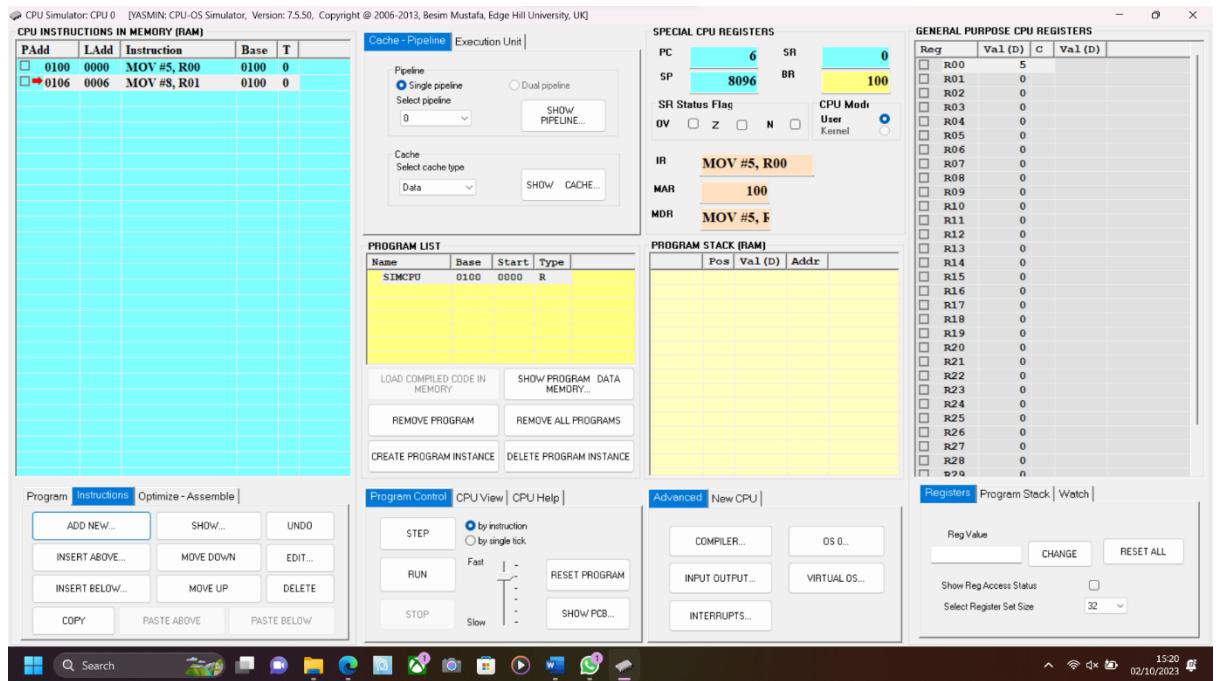
1. Create an instruction, which moves number 5 to register R00.



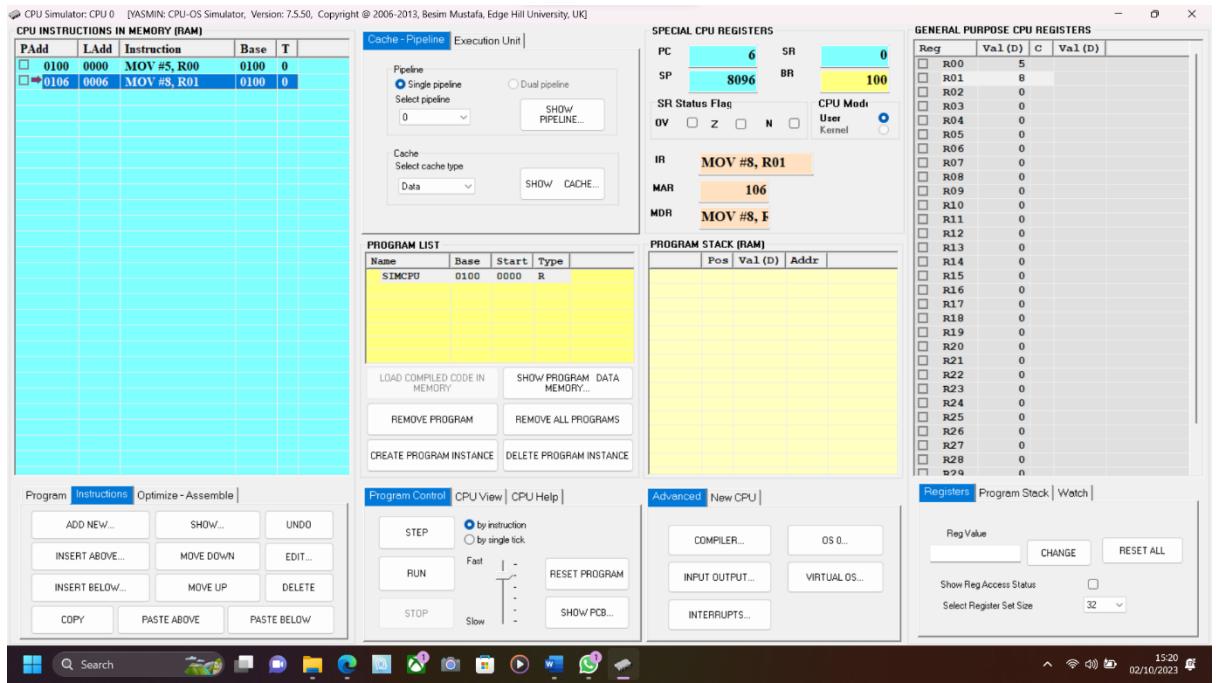
2. Execute the above instruction (to do this simply double click on it in the **Instruction Memory View**). Observe the result in the **CPU Registers** view (Image 4).



3. Create an instruction, which moves number 8 to register R01.



4. Execute it (You do this by double-clicking on the instruction).



5. Observe the contents of R00 and R01 in the **CPU Registers** view (Image 4).

GENERAL PURPOSE CPU REGISTERS			
Reg	Val (D)	C	Val (D)
R00	5		
R01	8		
R02	0		
R03	0		
R04	0		
R05	0		
R06	0		
R07	0		
R08	0		
R09	0		
R10	0		
R11	0		
R12	0		
R13	0		
R14	0		
R15	0		
R16	0		
R17	0		
R18	0		
R19	0		
R20	0		
R21	0		
R22	0		
R23	0		
R24	0		
R25	0		
R26	0		
R27	0		
R28	0		
R29	0		

6. Create an instruction, which adds the contents of R00 and R01.

The screenshot shows the YASMIN CPU Simulator interface. In the top left, the **CPU INSTRUCTIONS IN MEMORY (RAM)** window displays three instructions:

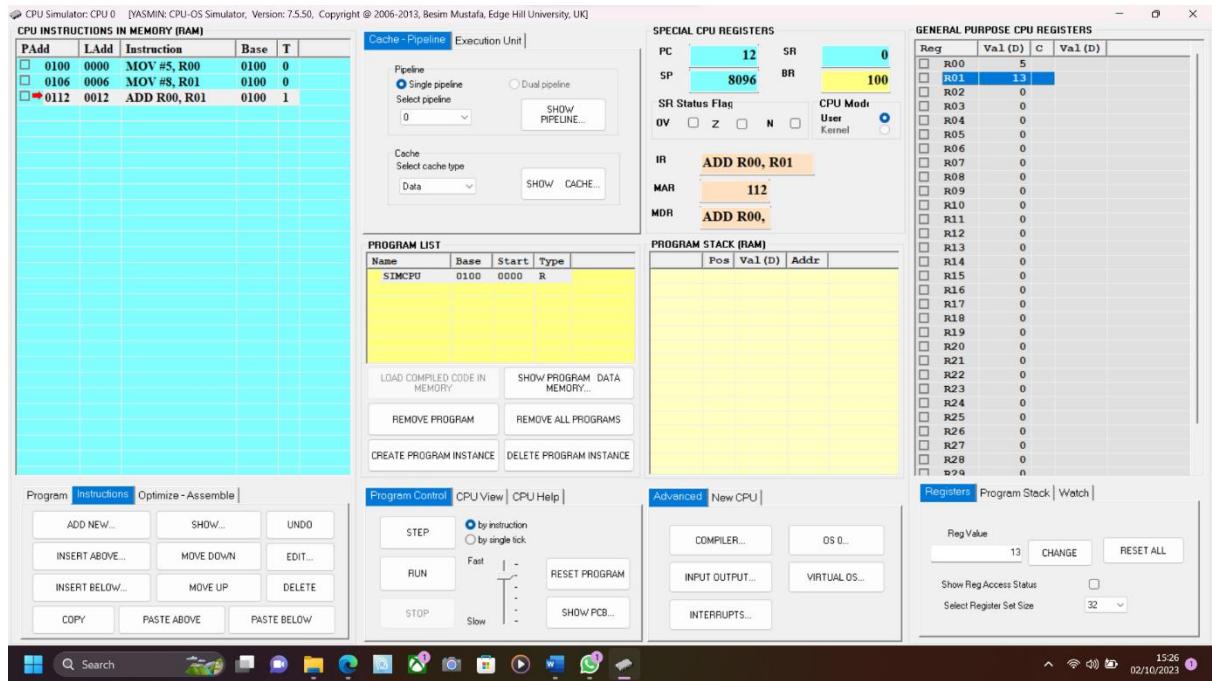
PAdd	LAdd	Instruction	Base	T
0100	0000	MOV #5, R00	0100	0
0106	0006	MOV #8, R01	0100	0
0112	0012	ADD R00, R01	0100	1

In the center, the **Execution Unit** window shows the pipeline stages for the ADD instruction. The PC stage has value 12, SR has value 0, SP has value 8096, and BR has value 100. The IR shows the instruction **MOV #8, R01**. The MAR and MDR both show **MOV #8, F**.

On the right, the **SPECIAL CPU REGISTERS** and **GENERAL PURPOSE CPU REGISTERS** windows are visible. The special registers show PC=12, SR=0, SP=8096, and BR=100. The general purpose registers show R00=5 and R01=8.

At the bottom, the **Program Control** window includes buttons for STEP, RUN, and STOP, along with options for COMPILER, INPUT OUTPUT, VIRTUAL OS, and INTERRUPTS.

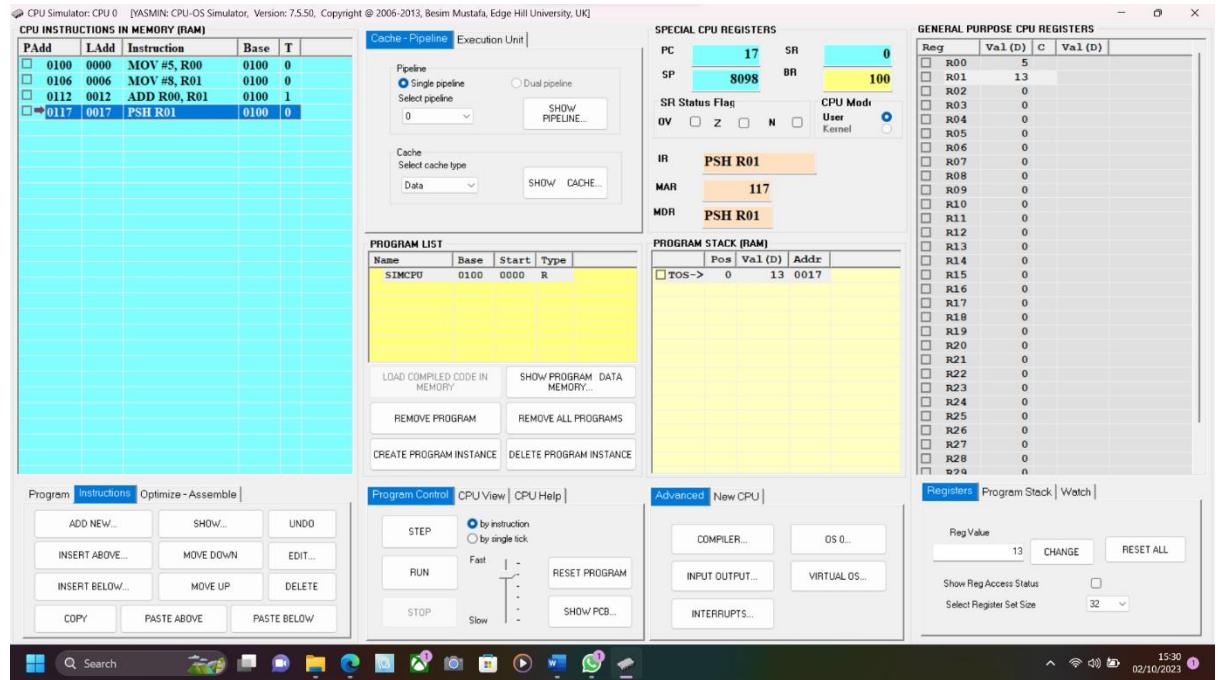
## 7. Execute it.



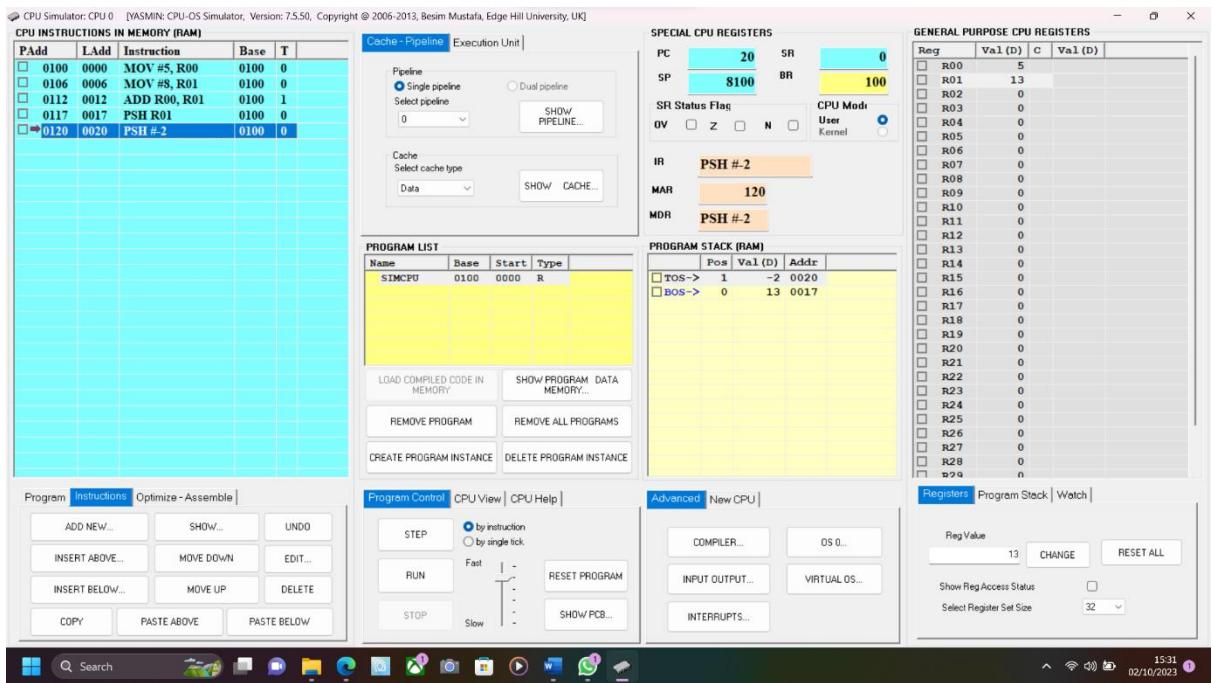
## 8. Observe which register the result is put in.

GENERAL PURPOSE CPU REGISTERS			
Reg	Val (D)	C	Val (D)
R00	5		
R01	13		
R02	0		
R03	0		
R04	0		
R05	0		
R06	0		
R07	0		
R08	0		
R09	0		
R10	0		
R11	0		
R12	0		
R13	0		
R14	0		
R15	0		
R16	0		
R17	0		
R18	0		
R19	0		
R20	0		
R21	0		
R22	0		
R23	0		
R24	0		
R25	0		
R26	0		
R27	0		
R28	0		
R29	0		

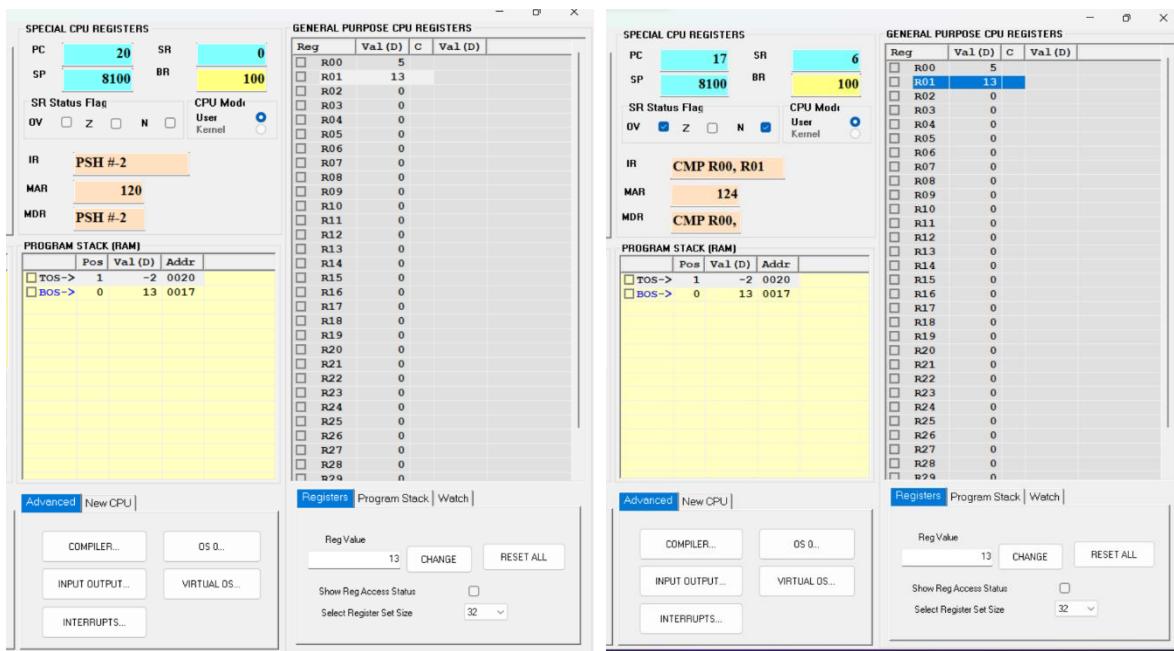
9. Create an instruction, which pushes the above result to the top of the hardware stack, and then execute it.



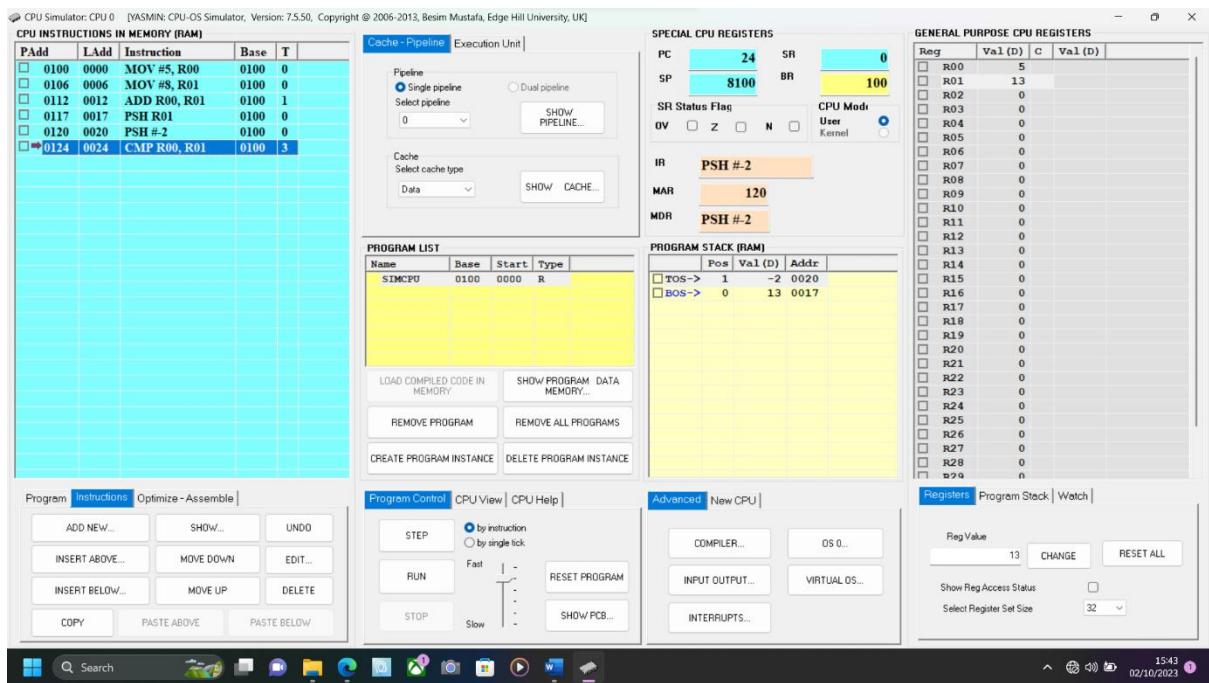
10. Create an instruction to push number-2 on top of the stack and execute it. Observe the value in **Program Stack** (Image 5).



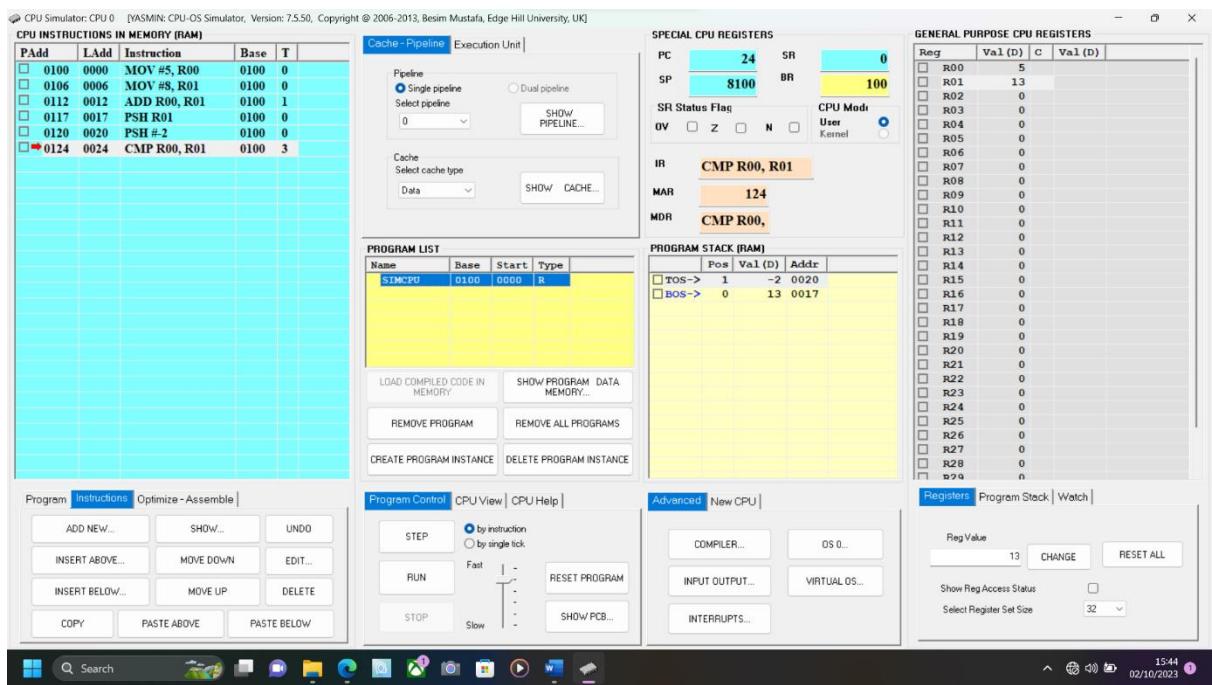
11. Observe the value in the **SP** register (**Special CPU Registers** view – Image 3). Whenever you push a value on **Program Stack**, the **SP** register is updated.



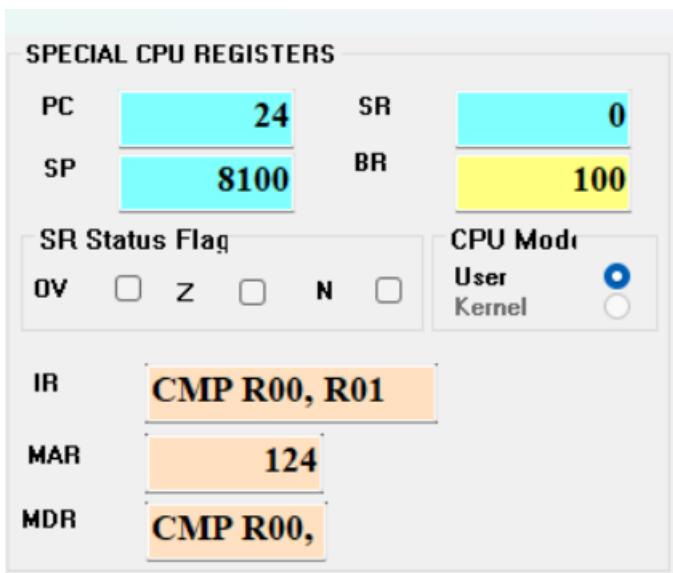
12. Create an instruction to compare the values in registers R00 and R01.



### 13. Execute it.



### 14. Observe the value in the SR register (Special CPU Registers view—Image 3).



15. Observe the status of the OV/Z/N parts of the status register. Which boxes are checked and which are not? What do they indicate?

SPECIAL CPU REGISTERS			
PC	24	SR	0
SP	8100	BR	100
SR Status Flag		CPU Mode	
OV	<input type="checkbox"/>	Z	<input type="checkbox"/>
N	<input type="checkbox"/>		<input type="radio"/> User
			<input type="radio"/> Kernel
IR	CMP R00, R01		
MAR	124		
MDR	CMP R00,		

SPECIAL CPU REGISTERS			
PC	24	SR	4
SP	8100	BR	100
SR Status Flag		CPU Mode	
OV	<input checked="" type="checkbox"/>	Z	<input type="checkbox"/>
N	<input type="checkbox"/>		<input type="radio"/> User
			<input type="radio"/> Kernel
IR	CMP R00, R01		
MAR	124		
MDR	CMP R00,		

SPECIAL CPU REGISTERS			
PC	24	SR	1
SP	8100	BR	100
SR Status Flag		CPU Mode	
OV	<input type="checkbox"/>	Z	<input checked="" type="checkbox"/>
N	<input type="checkbox"/>		<input type="radio"/> User
			<input type="radio"/> Kernel
IR	CMP R00, R01		
MAR	124		
MDR	CMP R00,		

SPECIAL CPU REGISTERS			
PC	24	SR	2
SP	8100	BR	100
SR Status Flag		CPU Mode	
OV	<input type="checkbox"/>	Z	<input type="checkbox"/>
N	<input checked="" type="checkbox"/>		<input type="radio"/> User
			<input type="radio"/> Kernel
IR	CMP R00, R01		
MAR	124		
MDR	CMP R00,		

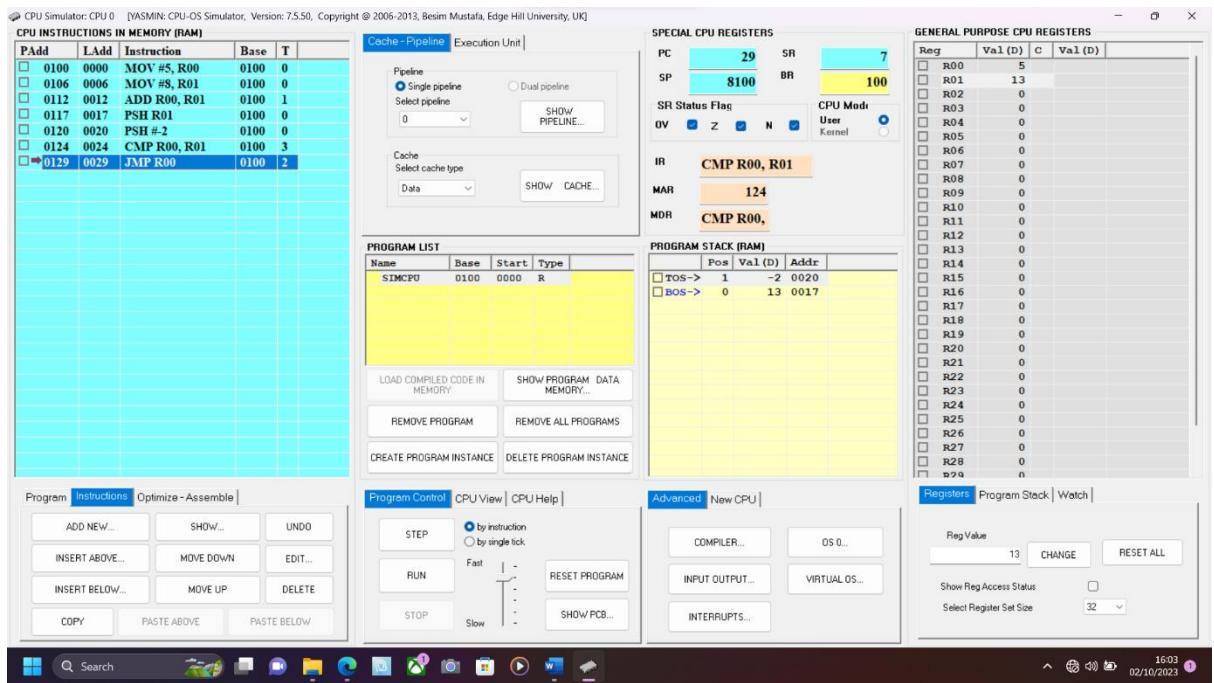
SPECIAL CPU REGISTERS			
PC	24	SR	5
SP	8100	BR	100
SR Status Flag		CPU Mode	
OV	<input checked="" type="checkbox"/>	Z	<input checked="" type="checkbox"/>
N	<input type="checkbox"/>		<input type="radio"/> User
			<input type="radio"/> Kernel
IR	CMP R00, R01		
MAR	124		
MDR	CMP R00,		

SPECIAL CPU REGISTERS			
PC	24	SR	6
SP	8100	BR	100
SR Status Flag		CPU Mode	
OV	<input checked="" type="checkbox"/>	Z	<input type="checkbox"/>
N	<input checked="" type="checkbox"/>		<input type="radio"/> User
			<input type="radio"/> Kernel
IR	CMP R00, R01		
MAR	124		
MDR	CMP R00,		

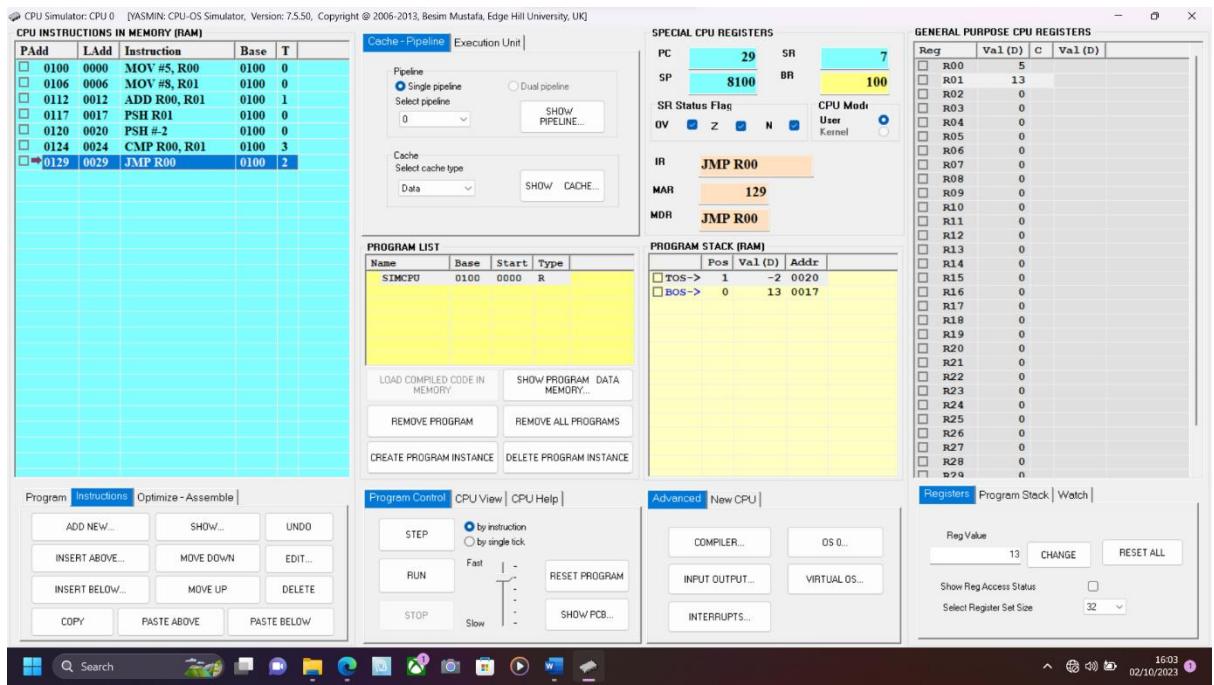
SPECIAL CPU REGISTERS			
PC	17	SR	3
SP	8100	BR	100
SR Status Flag		CPU Mode	
OV	<input type="checkbox"/>	Z	<input checked="" type="checkbox"/>
N	<input checked="" type="checkbox"/>		<input type="radio"/> User
			<input type="radio"/> Kernel
IR	CMP R00, R01		
MAR	124		
MDR	CMP R00,		

SPECIAL CPU REGISTERS			
PC	17	SR	7
SP	8100	BR	100
SR Status Flag		CPU Mode	
OV	<input checked="" type="checkbox"/>	Z	<input checked="" type="checkbox"/>
N	<input checked="" type="checkbox"/>		<input type="radio"/> User
			<input type="radio"/> Kernel
IR	CMP R00, R01		
MAR	124		
MDR	CMP R00,		

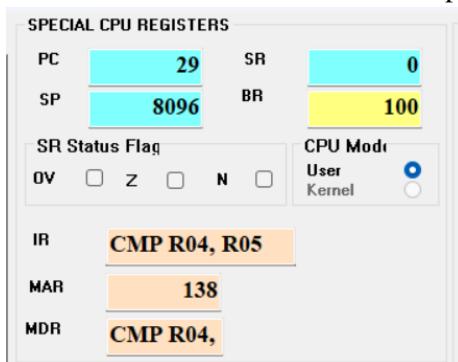
## 16. Create an instruction to unconditionally jump to the first instruction.



## 17. Execute it.



18. Observe the value in the **PC** register. This is the address of the next instruction to be executed. Make a note of which instruction it is pointing to?



19. Observe the values in the **PAdd** and **LAdd** columns. What do these values indicate? Are they different (**Hint:** Check out the Base Address value)?

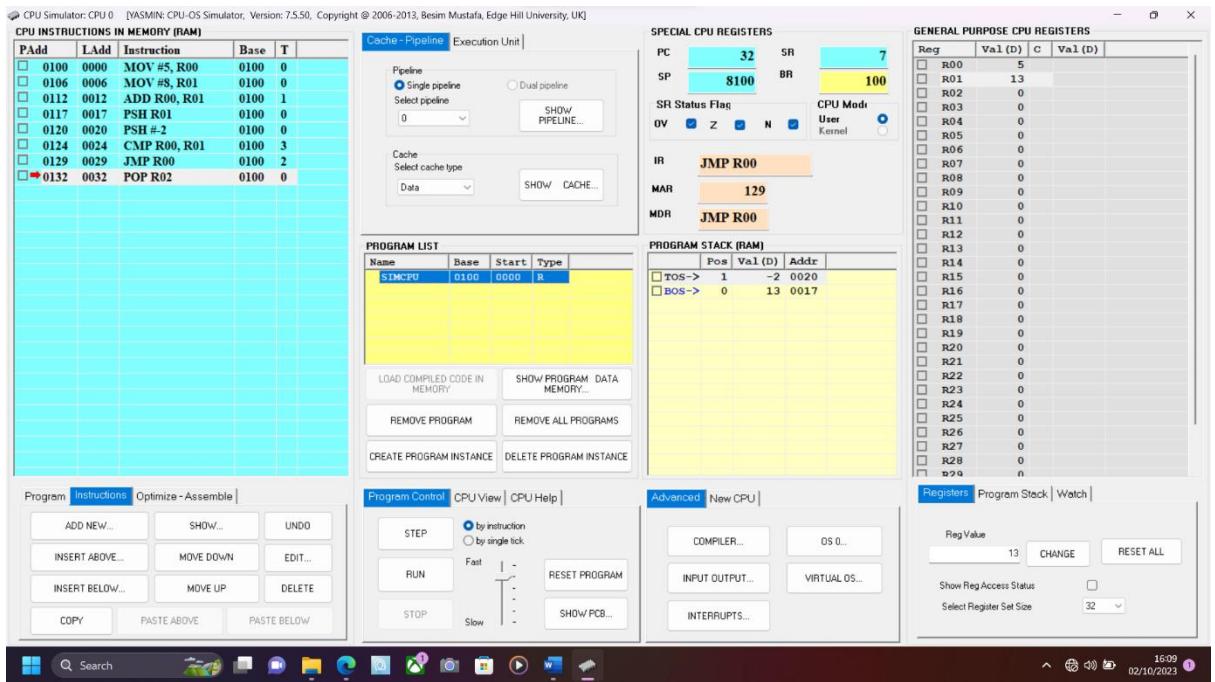
CPU INSTRUCTIONS I	
PAdd	LAdd
0100	0000
0106	0006
0112	0012
0117	0017
0120	0020
0124	0024
0129	0029

20. What is the difference between the **LAdd** value of the first instruction and the **LAdd** value of the second instruction? What does this value indicate (**Hint:** Think of the instruction lengths in bytes)?

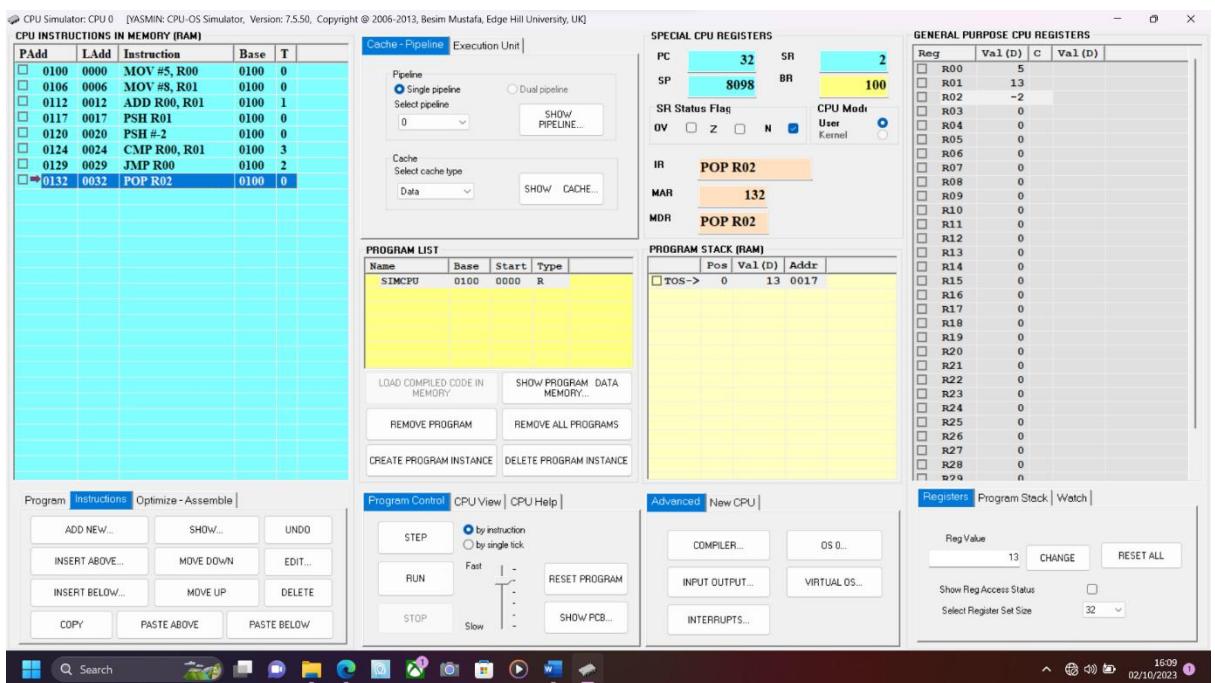
**Jawab:** Nilai Ladd pertama dan kedua berbeda karena nilainya ditambah 6

CPU INSTRUCTIONS I	
PAdd	LAdd
0100	0000
0106	0006

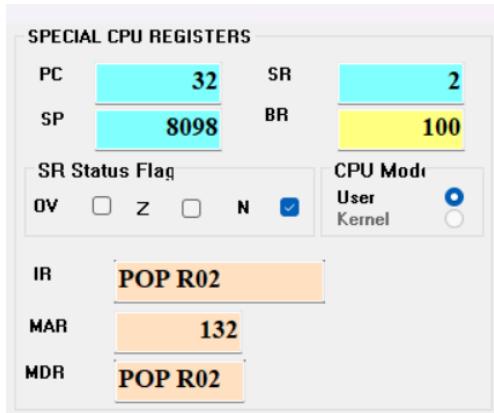
## 21. Create an instruction to pop the value on top of the Program Stack into register R02.



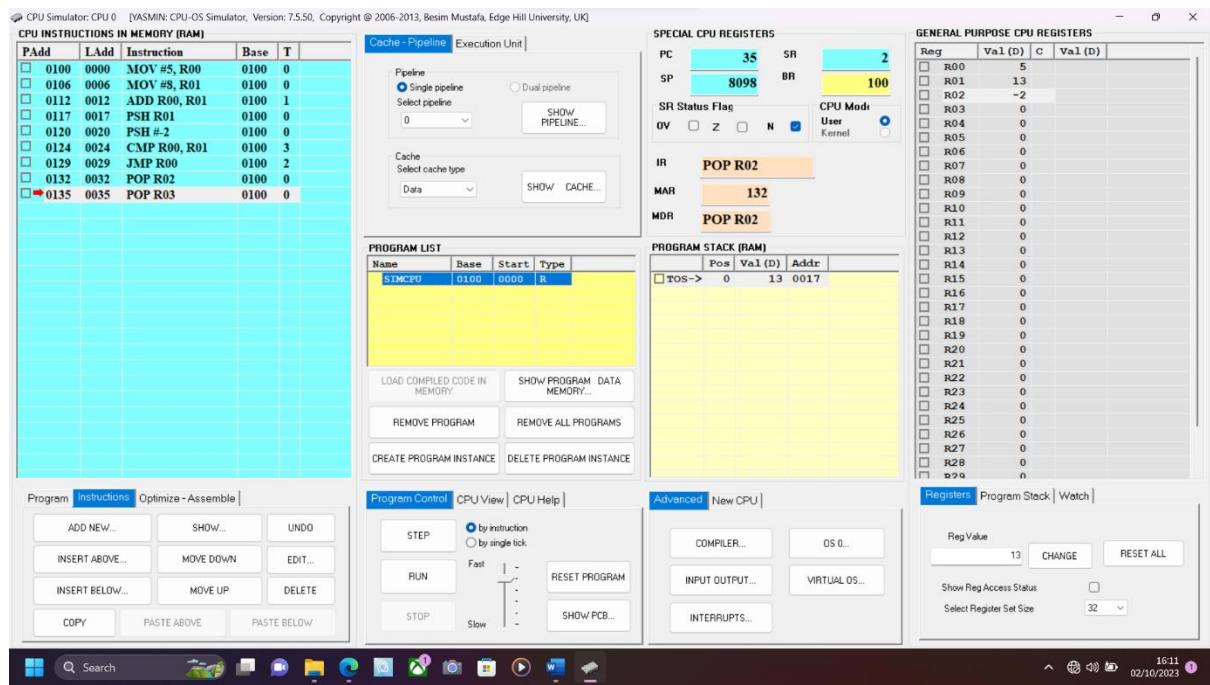
## 22. Execute it.



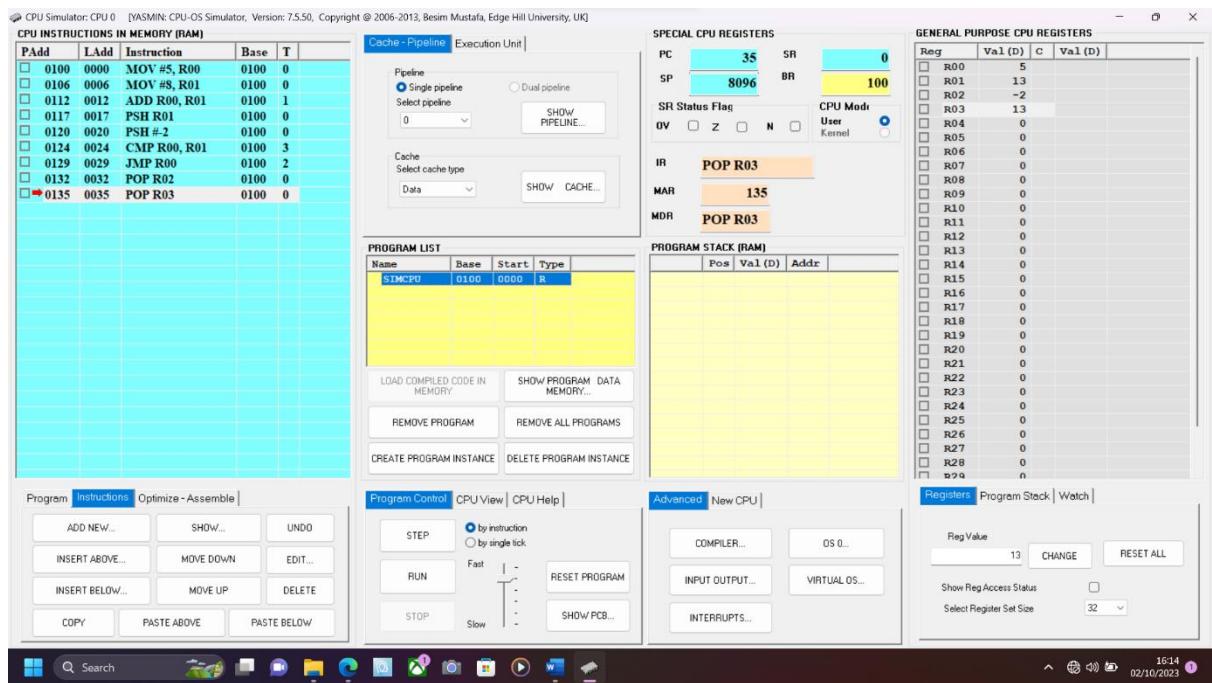
23. Observe the value in the **SP** register.



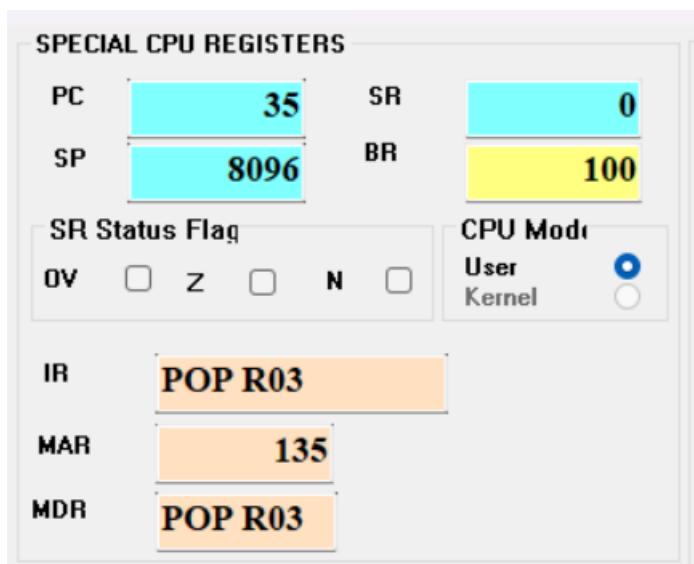
24. Create an instruction to **pop** the value on top of the **Program Stack** into register R03.



## 25. Execute it.

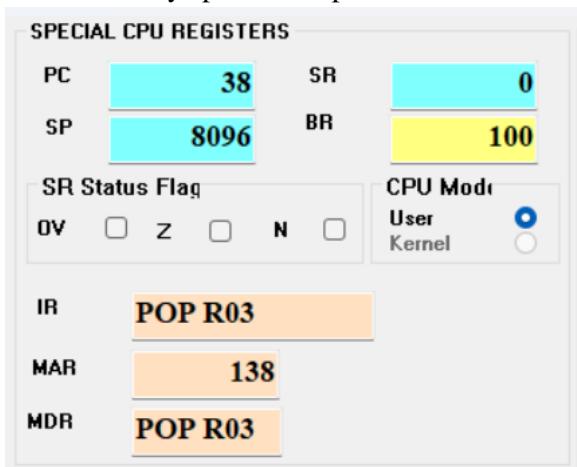


## 26. Observe the value in the SP register.

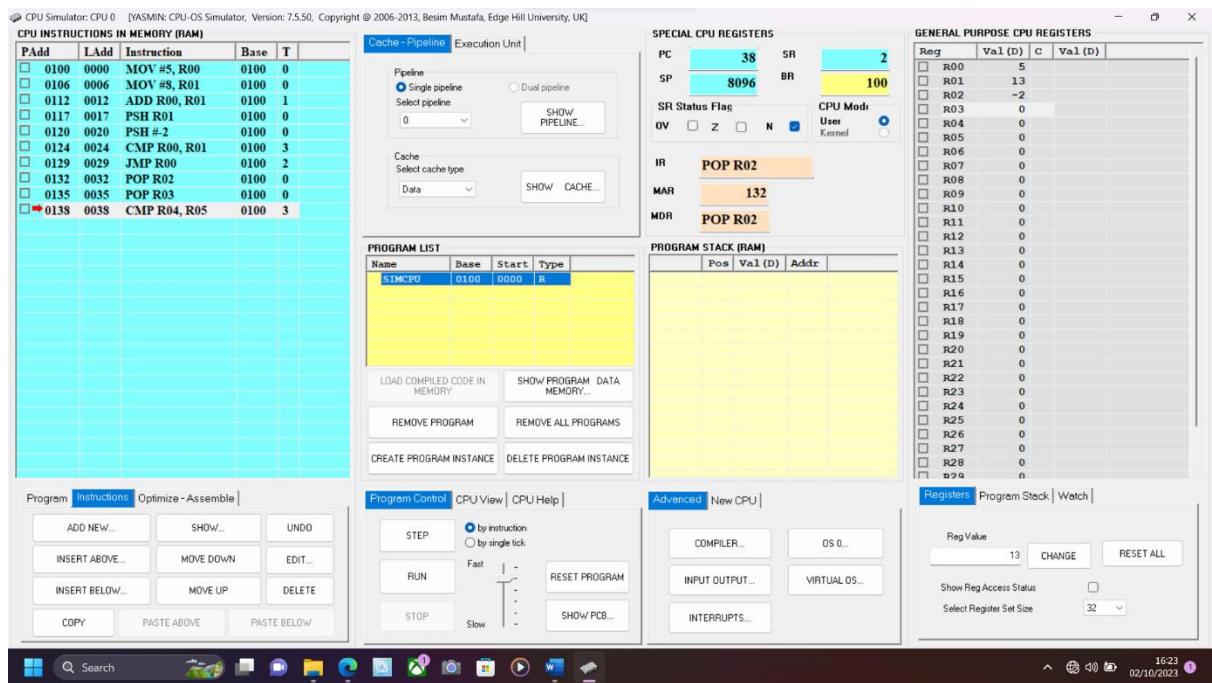


27. Execute the last instruction again. What happened? Explain.

**Jawab:** Adanya perubahan pada PC dan MAR



28. Create a compare instruction, which compares values in registers R04 and R05.



29. Manually insert two equal values in registers R04 and R05 (Image 4).

GENERAL PURPOSE CPU REGISTERS			
Reg	Val (D)	C	Val (D)
R00	5		
R01	13		
R02	-2		
R03	0		
R04	6		
R05	6		
R06	0		
R07	0		
R08	0		
R09	0		
R10	0		
R11	0		
R12	0		
R13	0		
R14	0		
R15	0		
R16	0		
R17	0		
R18	0		
R19	0		
R20	0		
R21	0		
R22	0		
R23	0		
R24	0		
R25	0		
R26	0		
R27	0		
R28	0		
R29	0		

30. One again execute the compare instruction in step 28 above.

The screenshot shows the YASMIN CPU-OS Simulator interface. On the left, the 'CPU INSTRUCTIONS IN MEMORY (RAM)' window displays assembly code. The highlighted instruction is:

```
0138 0038 CMP R04, R05    0100 3
```

In the center, the 'Execution Unit' panel shows the pipeline state. The PC is at 38, SP is at 8096, and the MAR is at 138. The MDR shows the result of the CMP instruction: 'CMP R04,'. The SR Status Flag register is also visible.

On the right, the 'GENERAL PURPOSE CPU REGISTERS' window shows the register values. Register R04 has been updated to 6, while R05 remains at 6. Other registers R00-R29 are at 0.

At the bottom, the 'Registers' panel shows the current value of R04 as 6, with options to change or reset it.

31. Which of the status flags OV/Z/N is set (i.e. box is checked)? Why?

**Jawab:** Z menandakan zero. Karena R04=R05,maka yang dicentang adalah Z (zero).

SPECIAL CPU REGISTERS	
PC	41
SP	8096
SR	1
BR	100

SR Status Flag: OV  Z  N

CPU Mode: User  Kernel

IR: CMP R04, R05

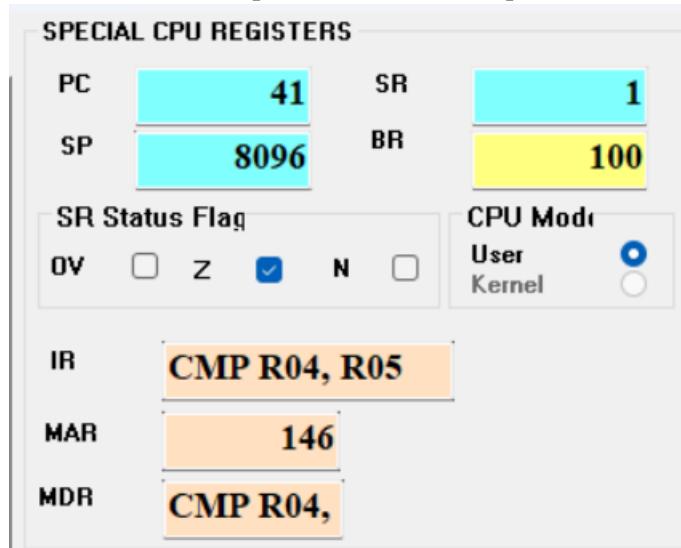
MAR: 141

MDR: CMP R04,

32. Manually insert a value in register R05 greater than that in register R04.

GENERAL PURPOSE CPU REGISTERS			
Reg	Val(D)	C	Val(D)
R00	5		
R01	13		
R02	-2		
R03	0		
R04	6		
<b>R05</b>	<b>10</b>		
R06	0		
R07	0		
R08	0		
R09	0		
R10	0		
R11	0		
R12	0		
R13	0		
R14	0		
R15	0		
R16	0		
R17	0		
R18	0		
R19	0		
R20	0		
R21	0		
R22	0		
R23	0		
R24	0		
R25	0		
R26	0		
R27	0		
R28	0		
R29	0		

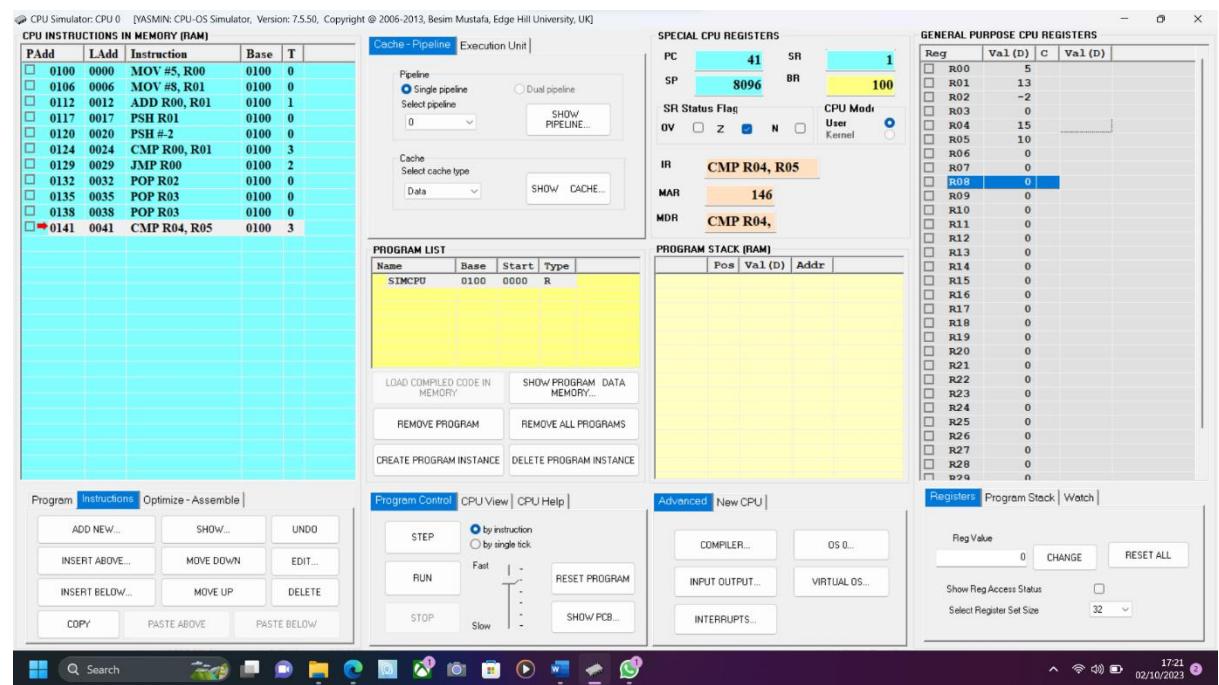
33. Execute the compare instruction in step 28 above.



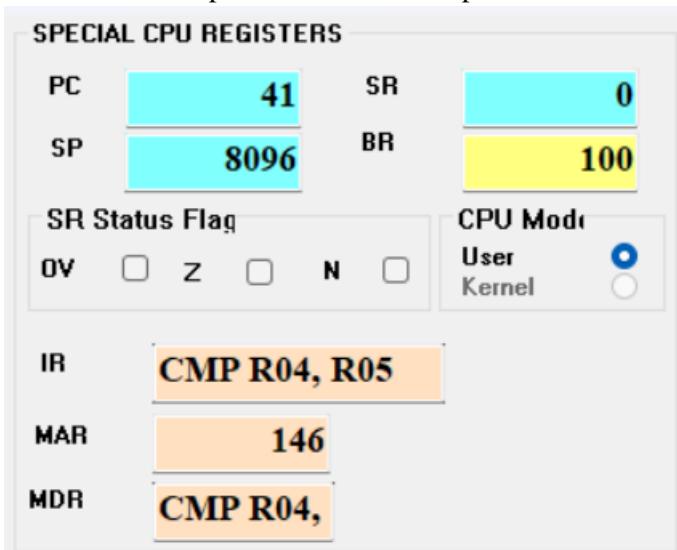
34. Which of the status flags OV/Z/N is set? Why?

**Jawab:** Status Nol tidak lagi dicentang karena nilai yang telah saya masukkan sudah memiliki perbandingan (positif).

35. Manually insert a value in register R04 greater than that in register R05.

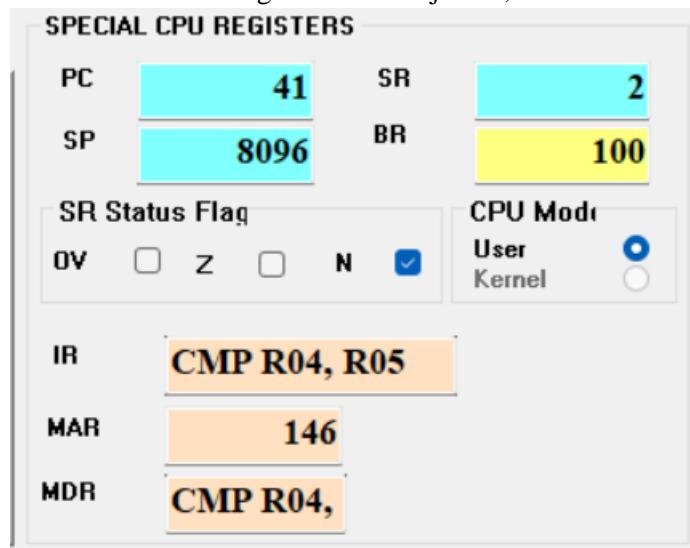


36. Execute the compare instruction in step 28 above.

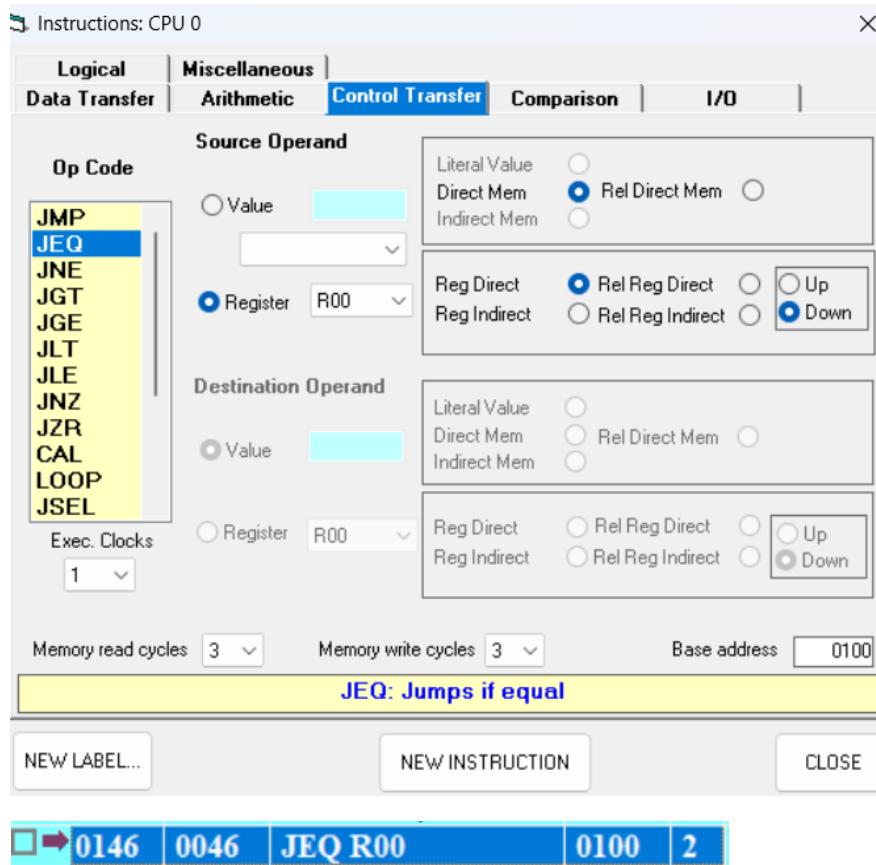


37. Which of the status flags OV/Z/N is set? Why?

Jawab: SR status flag berubah menjadi N, karena R04 lebih besar dari R05



38. Create an instruction, which will jump to the first instruction if the values in registers R04 and R05 are equal.



39. Test the above instruction by manually putting equal values in registers R04 and R05, then first executing the compare instruction followed by executing the jump instruction (**Remember:** You execute an instruction by double-clicking on it). Did it work?

40. Save the instructions in the **Instruction Memory View** in a file by clicking on the **SAVE...** button (Image 7).

