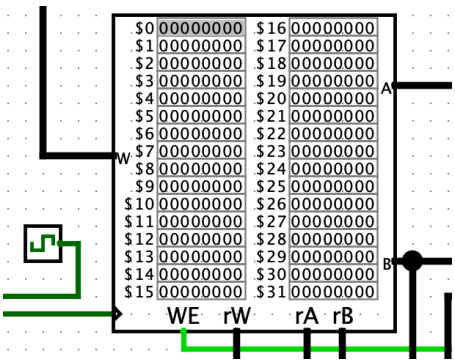
ทดสอบการทำงานของ CPU บน logicsim ว่าถูกต้องหรือไม่ (จาก File testcase.asm)

Address	Binary	Assembly
		main:
00000000	24080005	addiu $t0, $zero, 5
00000004	24090007	addiu $t1, $zero, 7
00000008	24100009	addiu $s0, $zero, 9
0000000c	24110002	addiu $s1, $zero, 2
00000010	0c000009	jal func_plus
00000014	00085040	sll $t2, $t0, 1
00000018	00085042	srl $t2, $t0, 1
0000001c	00085043	sra $t2, $t0, 1
00000020	08080000	j 0x200000
		func_plus:
00000024	01095021	addu $t2, $t0, $t1
00000028	02119021	addu $s2, $s0, $s1
0000002c	2d4b0014	sltiu $t3, $t2, 20
00000030	11600012	beq $t3, $zero, bistwise_2
00000034	294b0004	slti $t3, $t2, 4
00000038	01095023	subu $t2, $t0, $t1
0000003c	0148582b	sltu $t3, $t2, $t0
00000040	15600009	bne $t3, $zero, bistwise
00000044	0109582a	slt $t3, $t0, $t1
00000048	15600007	bne $t3, $zero, bistwise
		load_store:
0000004c	27bdfff8	addiu $sp, $sp, -8
00000050	afa80004	sw $t0, 4($sp)
00000054	afa90000	sw $t1, 0($sp)
00000058	8fab0004	lw $t3, 4($sp)
0000005c	8fac0000	lw $t4, 0($sp)
00000060	27bd0008	addiu $sp, $sp, 8
00000064	0800001f	j bistwise_2
		bistwise:
00000068	01096024	and $t4, $t0, $t1
0000006c	01096825	or $t5, $t0, $t1
00000070	01097026	xor $t6, $t0, $t1
00000074	01097827	nor $t7, $t0, $t1
00000078	08000013	j load_store
		bistwise_2:
0000007c	310c000c	andi $t4, $t0, 12
00000080	350d000d	ori $t5, $t0, 13
00000084	390e000e	xori $t6, $t0, 14
00000088	03e00008	jr $ra

ทดสอบการทำงานใน logic sim

อธิบาย :

ค่าเริ่มต้นทุก registers มีค่าเป็น 0x000000

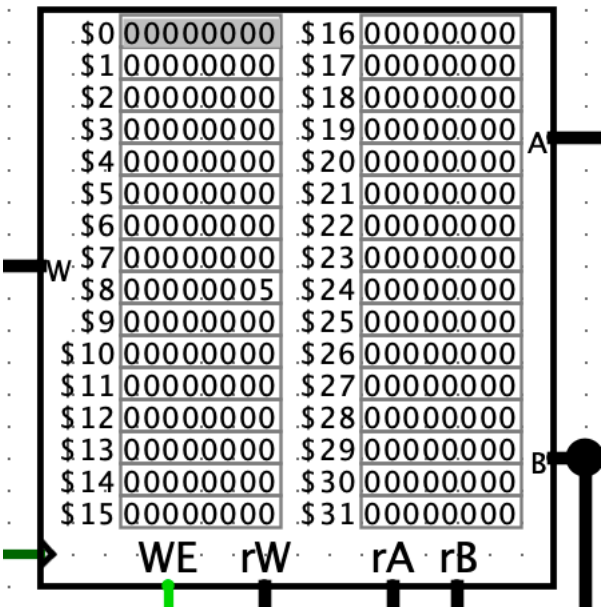


Main:

```
addiu   $t0, $zero, 5
```

อธิบาย :

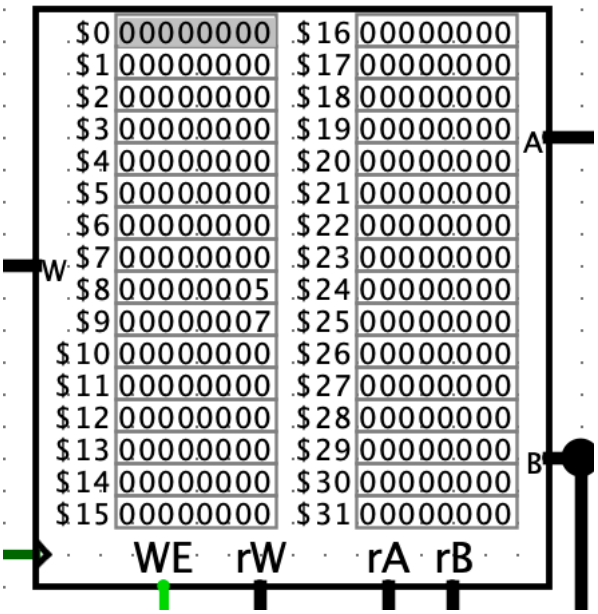
จากคำสั่งข้างต้น จะนำค่า 0 มาบวกกับค่าคงที่ 5 มาใส่ที่ $t0 ดังนั้น register $8 จึงเท่ากับ 0x000005




```
addiu   $t1, $zero, 7
```

อธิบาย :

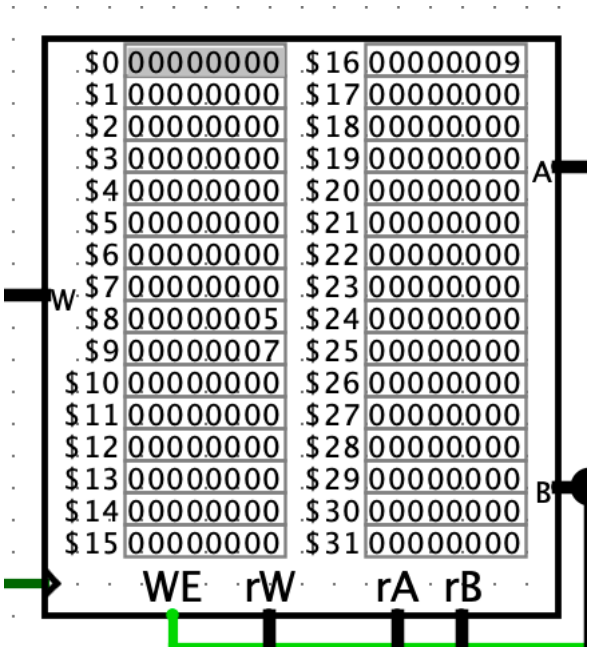
จากคำสั่งข้างต้น จะนำค่า 0 มาบวกกับค่าคงที่ 7 มาใส่ที่ $t1 ดังนั้น register $9 จึงเท่ากับ 0x000007



```
addiu   $s0, $zero, 9
```

อธิบาย :

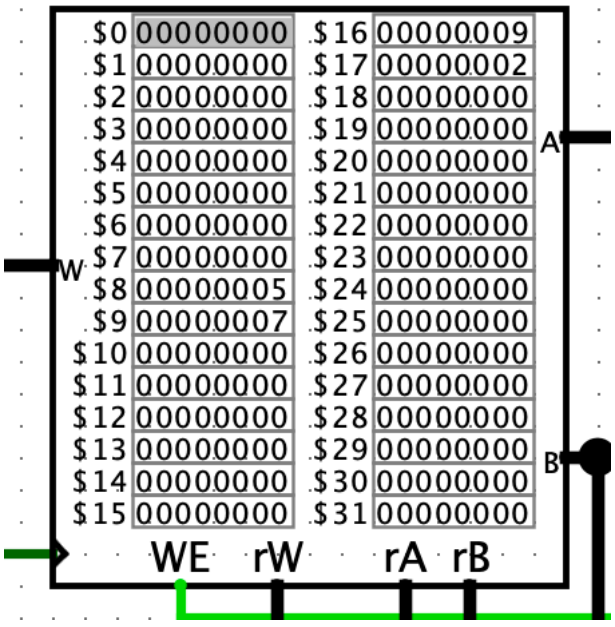
จากคำสั่งข้างต้น จะนำค่า 0 มาบวกกับค่าคงที่ 9 (แบบ unsigned) มาใส่ที่ $s0 ดังนั้น register $16 จึงเท่ากับ 0x00009




```
addiu   $s1, $zero, 2
```

อธิบาย :

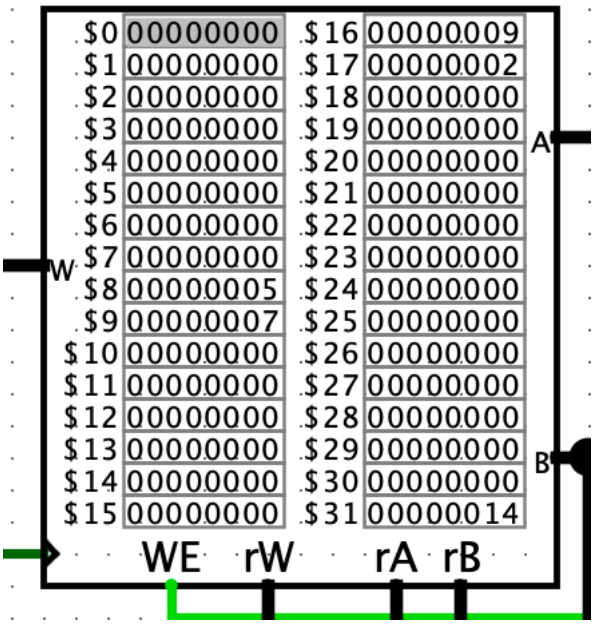
จากคำสั่งข้างต้น จะนำค่า 0 มาบวกกับค่าคงที่ 2 (แบบ unsigned) มาใส่ที่ $s1 ดังนั้น register $17 จึงเท่ากับ 0x00002



```
jal func_plus
```

อธิบาย :

จากคำสั่งข้างต้น จะโดดไปยังคำสั่ง func_plus และเก็บ address ไว้ที่ $ra ดังนั้น register $31 จึงเท่ากับ 000014 ดังรูป

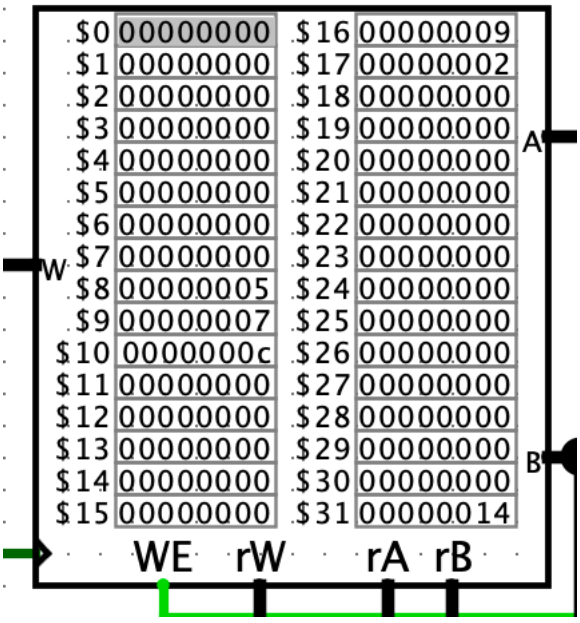


Func_plus:

```
addu $t2, $t0, $t1
```

อธิบาย :

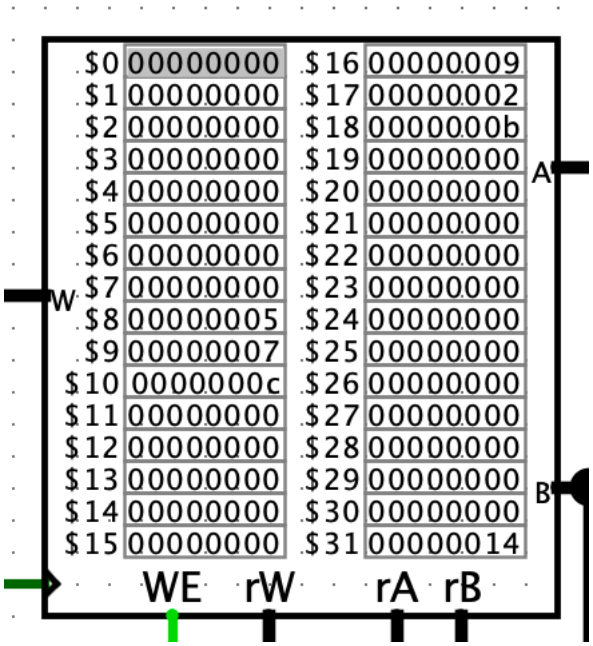
จากคำสั่งข้างต้น จะนำค่าใน $t0 มาบวกกับค่าใน $t1 (แบบ unsigned) แล้วนำผลลัพธ์มาใส่ที่ register $t2 ดังนั้น register $10 จึงเท่ากับ 0x00000c



```
addu $s2, $s0, $s1
```

อธิบาย :

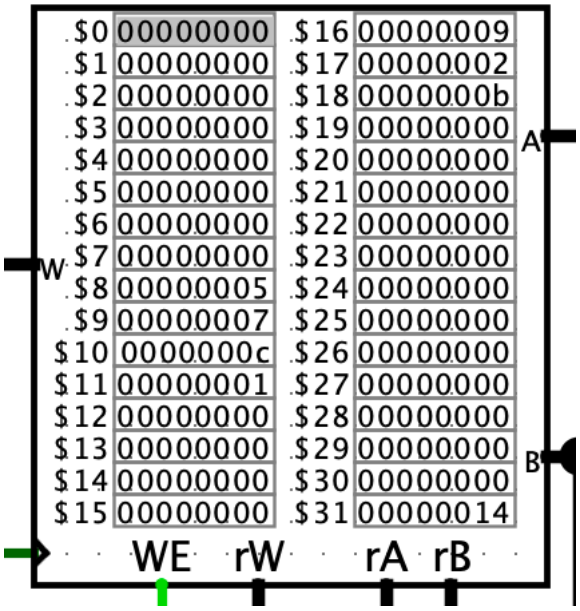
จากคำสั่งข้างต้น จะนำค่าใน $s0 มาบวกกับ ค่าใน $s1 (แบบ unsigned) แล้วนำผลลัพธ์มาใส่ที่ register $s2 ดังนั้น register $18 จึงเท่ากับ 0x0000b




```
sltiu $t3, $t2, 20
```

อธิบาย :

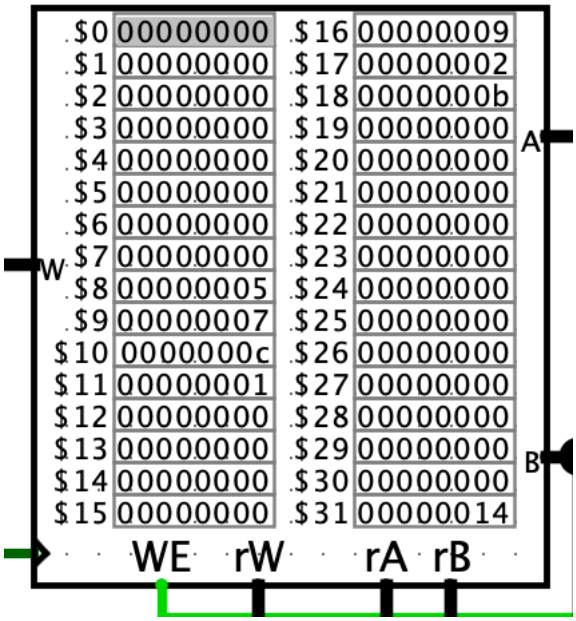
จากคำสั่งข้างต้น จะนำค่าใน $t2 มาเทียบกับ ค่าคงที่ 20 ว่าค่าใน register $t2 น้อยกว่าหรือไม่ (แบบ unsigned) ถ้าเป็นจริงผลลัพธ์จะมีค่าเท่ากับ 1 แต่ถ้าเป็นเท็จผลลัพธ์จะมีค่าเท่ากับ 0 แล้วนำผลลัพธ์ที่ได้มาใส่ที่ register $t3 ดังนั้น register $11 จึงมีค่าเท่ากับ 0x00001



```
beq $t3, $zero, bistwise_2
```

อธิบาย :

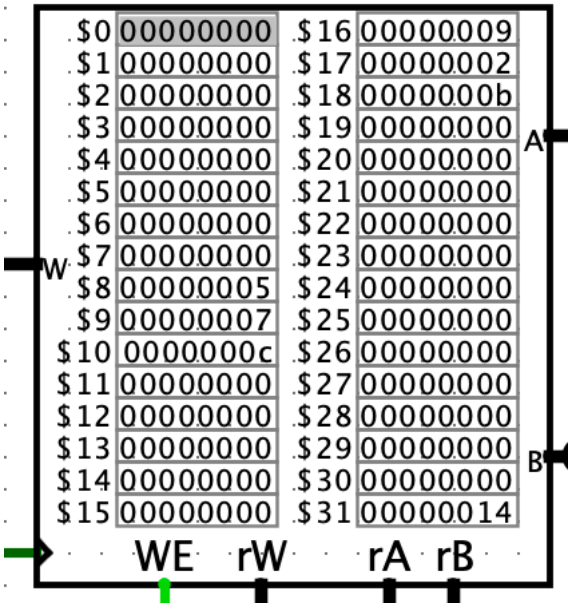
จากคำสั่งข้างต้น จะนำค่าใน $t3 มาเทียบกับ $0 ว่าค่าใน register $t3 มีค่าเท่ากับ 0 หรือไม่ ถ้าเท่ากับ 0 จะกระโดดไปทำที่ bistwise แต่ถ้าไม่เท่ากับ 0 จะทำบรรทัดถัดไปซึ่งในที่นี้ $11 มีค่าเท่ากับ 0x00001 ทำให้ทำบรรทัดต่อไป




```
slti $t3, $t2, 4
```

อธิบาย :

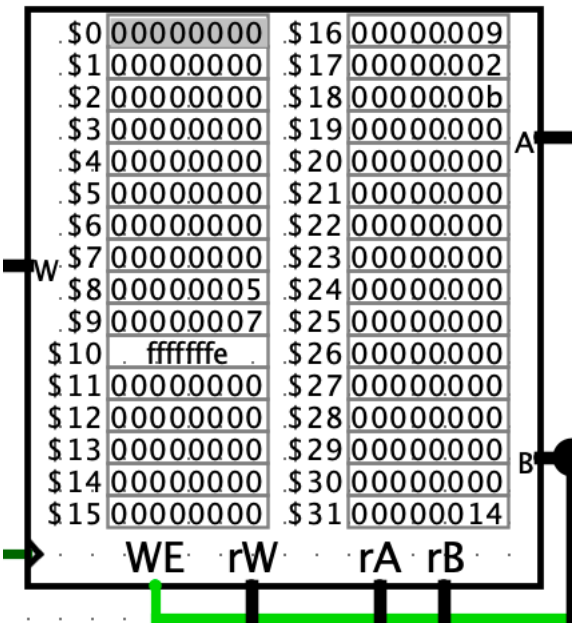
จากคำสั่งข้างต้น จะนำค่าใน $t2 มาเทียบกับ ค่าคงที่ 4 ว่าค่าใน register $t2 น้อยกว่าหรือไม่ ถ้าเป็นจริงผลลัพธ์จะมีค่าเท่ากับ 1 แต่ถ้าเป็นเท็จผลลัพธ์จะมีค่าเท่ากับ 0 แล้วนำผลลัพธ์ที่ได้มาใส่ที่ register $11 ดังนั้น register $11 จึงมีค่าเท่ากับ 0x00000 ทำให้ทำบรรทัดต่อไป



```
subu $t2, $t0, $t1
```

อธิบาย :

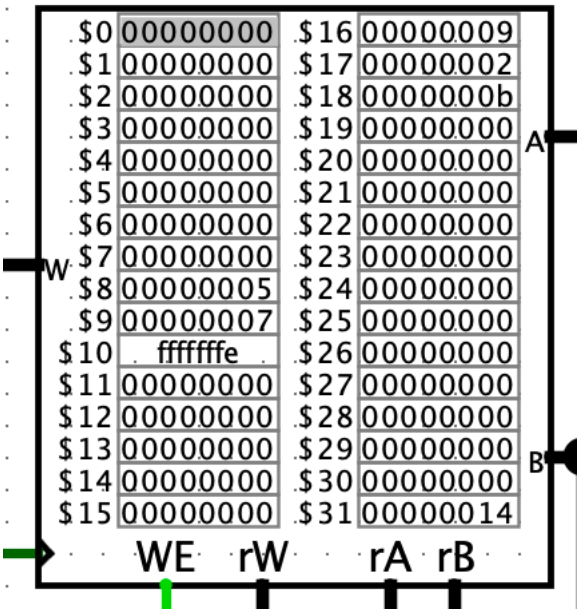
จากคำสั่งข้างต้น จะนำค่าใน $t0 มาลบกับค่าใน $t1 (แบบ unsigned) แล้วนำผลลัพธ์มาใส่ที่ register $t2 ดังนั้น register $10 จึงเท่ากับ 0xfffffe




```
sltu $t3, $t2, $t0
```

อธิบาย :

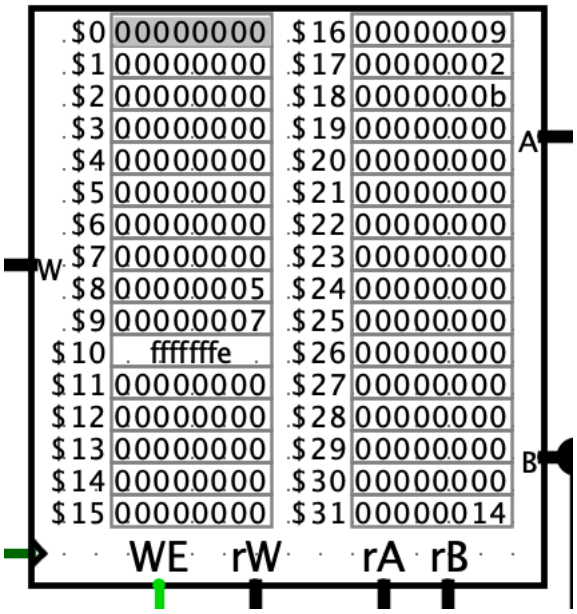
จากคำสั่งข้างต้น จะนำค่าใน $t2 มาเทียบกับ ค่าใน $t0 ว่าค่าใน register $t2 น้อยกว่าหรือไม่ (แบบ unsigned) ถ้าเป็นจริงผลลัพธ์จะมีค่าเท่ากับ 1 แต่ถ้าเป็นเท็จผลลัพธ์จะมีค่าเท่ากับ 0 แล้วนำผลลัพธ์ที่ได้มาใส่ที่ register $t3 ดังนั้น register $11 จึงมีค่าเท่ากับ 0x00000



```
bne $t3, $zero, bistwise
```

อธิบาย :

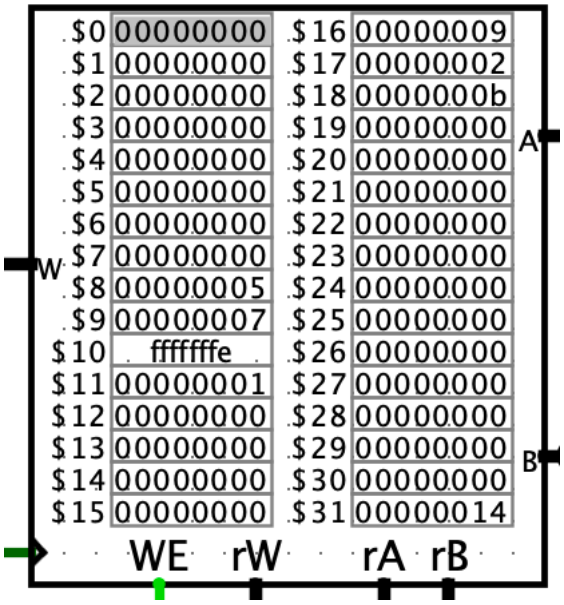
จากคำสั่งข้างต้น จะนำค่าใน $t3 มาเทียบกับ $0 ว่าค่าใน register $t3 มีค่าไม่เท่ากับ 0 หรือไม่ถ้าไม่เท่ากับ 0 จะกระโดดไปทำที่ bistwise แต่ถ้าเท่ากับ 0 จะทำบรรทัดถัดไป ซึ่งในที่นี้ $11 มีค่าเท่ากับ 0x00000




```
slt $t3, $t0, $t1
```

อธิบาย :

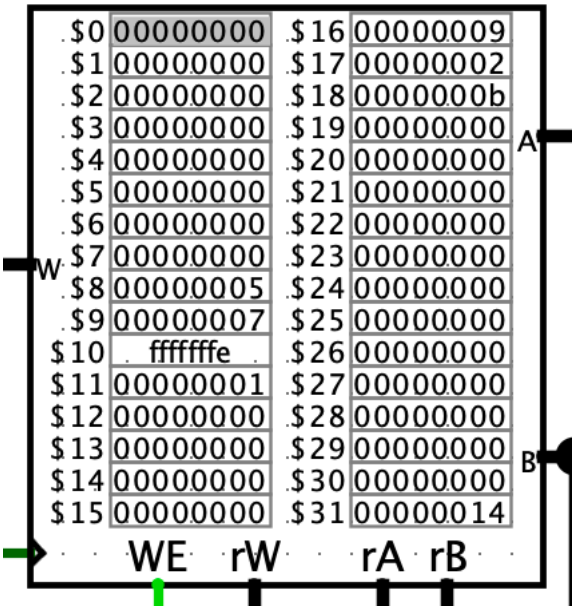
จากคำสั่งข้างต้น จะนำค่าใน $t0 มาเทียบกับ ค่าใน $t1 ว่าค่าใน register $t0 น้อยกว่าหรือไม่ ถ้าเป็นจริงผลลัพธ์จะมีค่าเท่ากับ 1 แต่ถ้าเป็นเท็จผลลัพธ์จะมีค่าเท่ากับ 0 แล้วนำผลลัพธ์ที่ได้มาใส่ที่ register $t3 ดังนั้น register $11 จึงมีค่าเท่ากับ 0x00001



```
bne $t3, $zero, bistwise
```

อธิบาย :

จากคำสั่งข้างต้น จะนำค่าใน $t3 มาเทียบกับ $0 ว่าค่าใน register $t3 มีค่าไม่เท่ากับ 0 หรือไม่ถ้าไม่เท่ากับ 0 จะกระโดดไปทำที่ bistwise แต่ถ้าเท่ากับ 0 จะทำบรรทัดถัดไป ซึ่งในที่นี้ $11 มีค่าเท่ากับ 0x000001

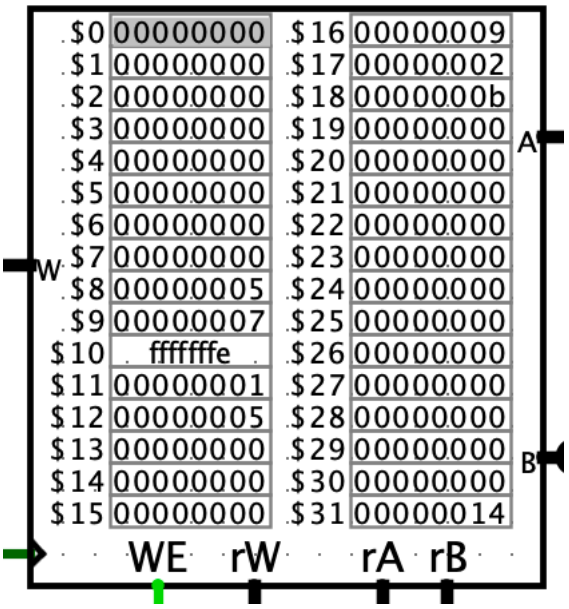


Bistwise:

```
and $t4, $t0, $t1
```

อธิบาย :

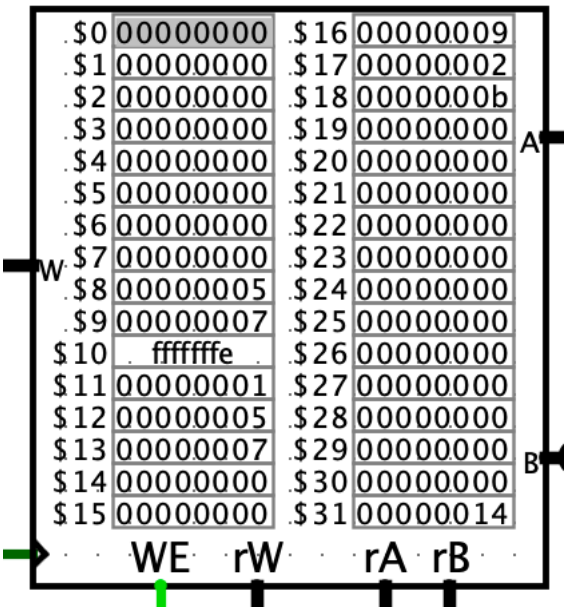
จากคำสั่งข้างต้น จะนำค่าใน $t0 และ $t1 มา AND กัน และนำ
ค่าไปใส่ไว้ที่ $t4 ดังนั้น Register $12 มีค่าเท่ากับ 0x000005



```
or $t5, $t0, $t1
```

อธิบาย :

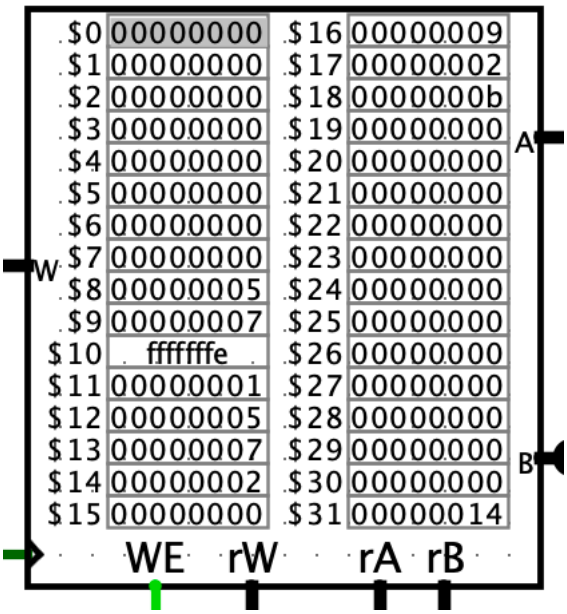
จากคำสั่งข้างต้น จะนำค่าใน $t0 และ $t1 มา OR กัน และนำค่าไปใส่ไว้ที่ $t3 ดังนั้น Register $12 มีค่าเท่ากับ 0x000007




```
xor $t6, $t0, $t1
```

อธิบาย :

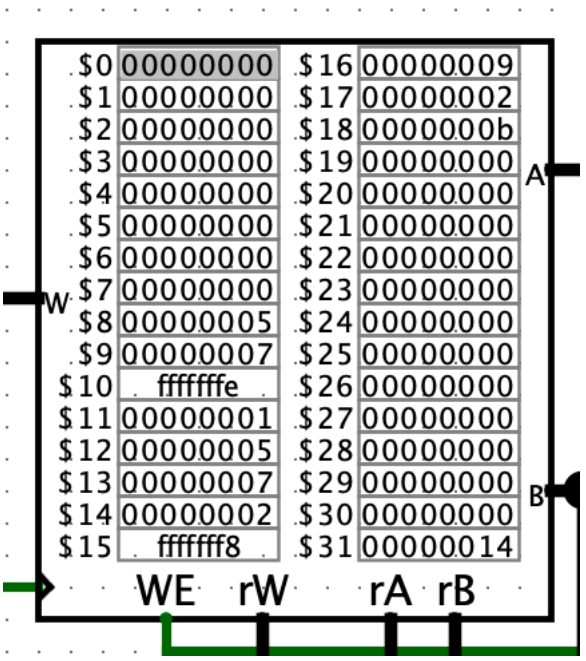
จากคำสั่งข้างต้น จะนำค่าใน $t0 และ $t1 มา XOR กัน และนำค่าไปใส่ไว้ที่ $t6 ดังนั้น Register $14 มีค่าเท่ากับ 0x000002



```
nor $t7, $t0, $t1
```

อธิบาย :

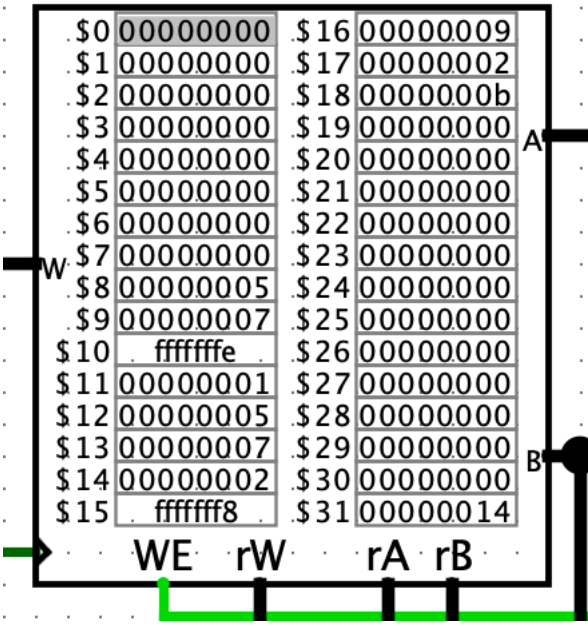
จากคำสั่งข้างต้น จะนำค่าใน $t0 และ $t1 มา NOR กัน และนำค่าไปใส่ไว้ที่ $t7 ดังนั้น Register $15 มีค่าเท่ากับ fxfffff8




```
j load_store
```

อธิบาย :

จากคำสั่งข้างต้น จะโดดไปยังคำสั่ง load_store แต่ไม่ได้เก็บ address ไว้ที่ $ra ดังนั้น register $31 จึงไม่เปลี่ยนแปลง ดังรูป

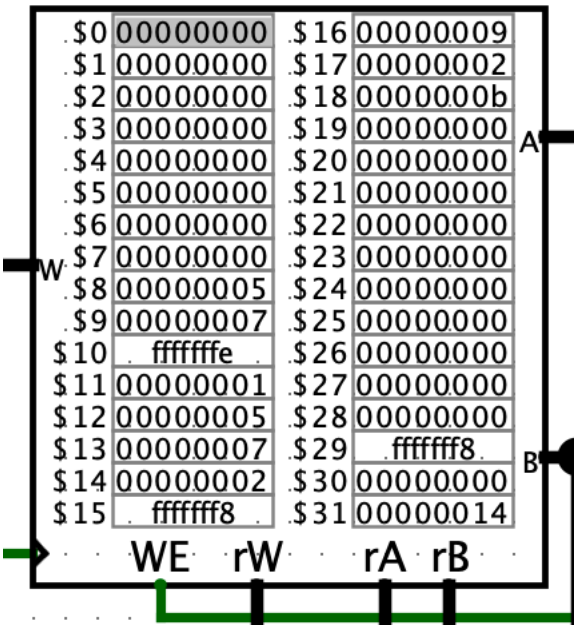


Load_store:

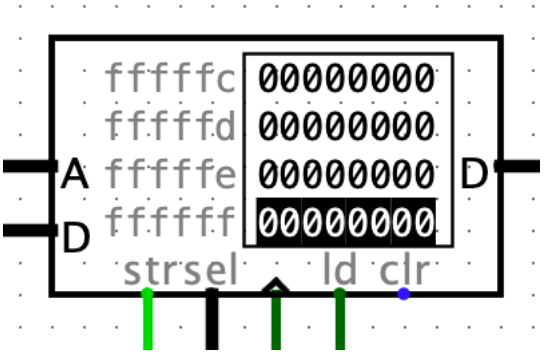
```
addiu $sp, $sp, -8
```

อธิบาย :

จากคำสั่งข้างต้นจะทำการนำค่า $sp มาบวกกับ -8 (แบบ unsigned) แล้วมาใส่ใน $sp เหมือนเดิม ดังนั้นค่า Register $29 จึงมีค่า เท่ากับ fffffff8



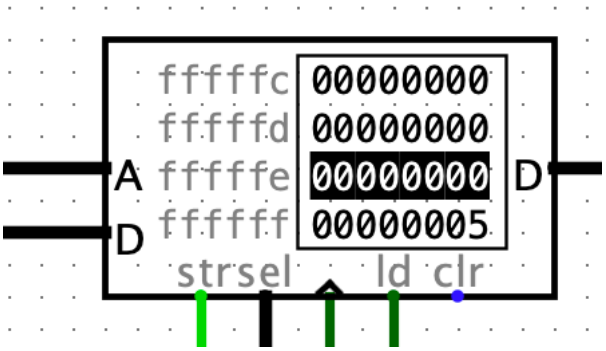
เพราะฉะนั้น ค่า address ของ 0($sp) ใน Memory คือ ffffff



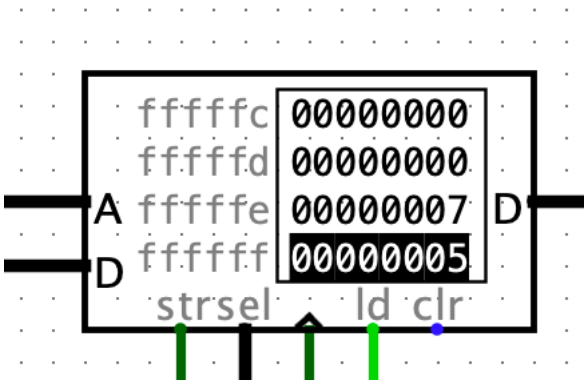

```
sw $t0, 4($sp)
```

อธิบาย :

จากคำสั่งข้างต้นจะทำการ store ค่า $t0 ไว้ที่ 4($sp) ซึ่งมี address ใน Memory เป็น ffffff ดังนั้นจึงเก็บค่า 0x000005 ไว้



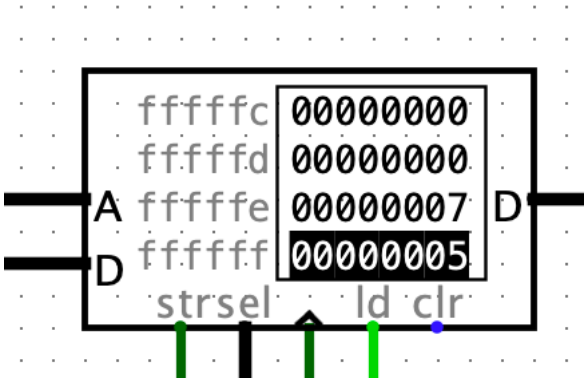
```
sw $t1, 0($sp)
```

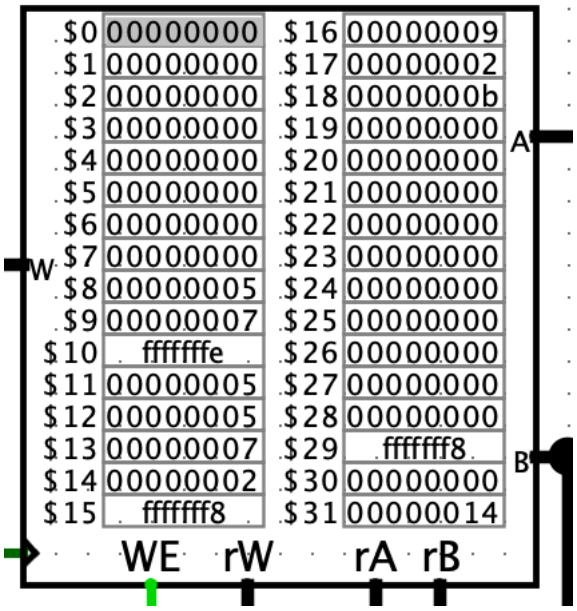


อธิบาย :

จากคำสั่งข้างต้นจะทำการ store ค่า $t1 ไว้ที่ 0 ($sp) ซึ่งมี address ใน Memory เป็น fffffe ดังนั้นจึงเก็บค่า 0x0007 ไว้

```
lw $t3, 4($sp)
```





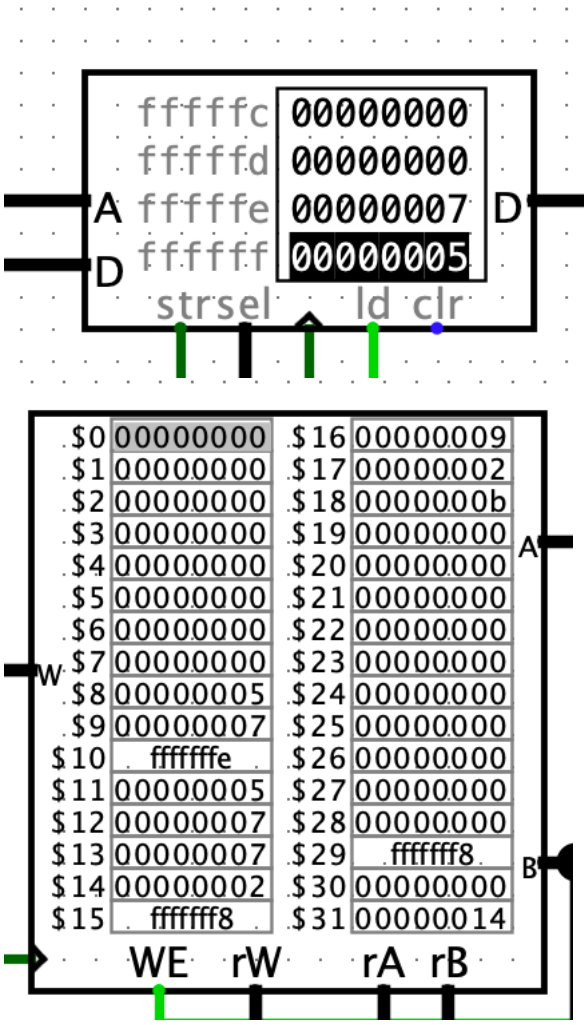
อธิบาย :

จากคำสั่งข้างต้นจะทำการ load ค่าจาก 4($sp) ซึ่งมี address ใน Memory เป็น ffffff มาไว้ที่ $t3 ดังนั้น Register $11 มีค่าเท่ากับ 0x000005

```
lw $t4, 0($sp)
```

อธิบาย :

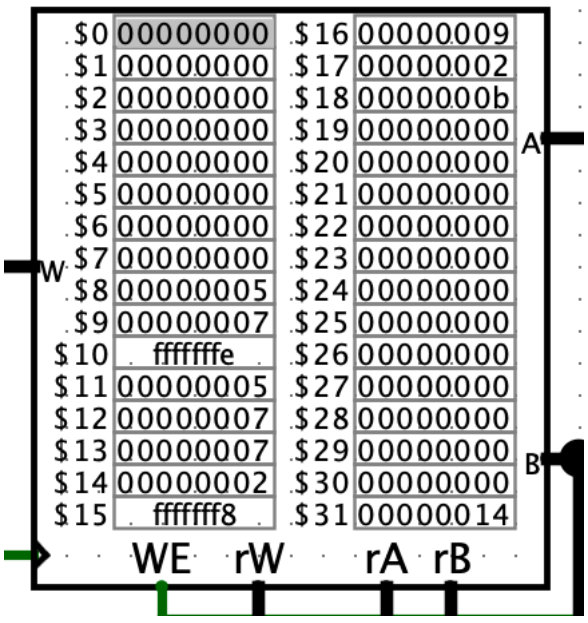
จากคำสั่งข้างต้นจะทำการ load ค่าจาก 0($sp) ซึ่งมี address ใน Memory เป็น fffffe มาไว้ที่ $t4 ดังนั้น Register $12 มีค่าเท่ากับ 0x0007




```
addiu $sp, $sp, 8
```

อธิบาย :

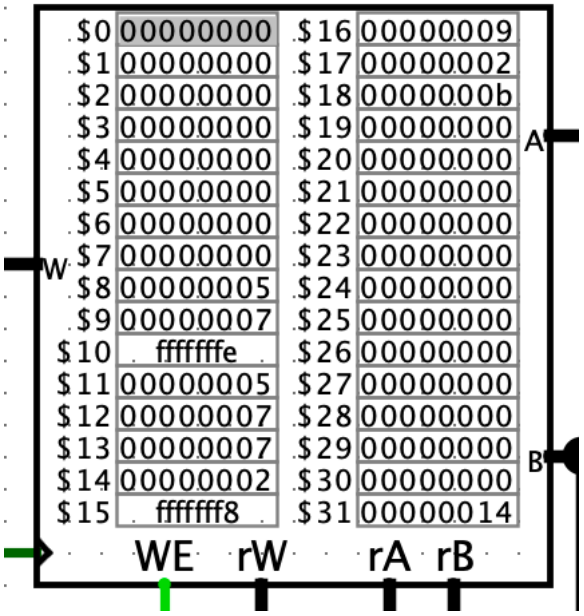
จากคำสั่งข้างต้นจะทำการนำค่า $sp ซึ่งมีค่าเท่ากับ ffffffff8 มาบวกกับ 8 (แบบ unsigned) แล้วมาใส่ใน $sp เหมือนเดิม ดังนั้นค่า Register $29 จึงมีค่า เท่ากับ 00000000



```
j bistwise_2
```

อธิบาย :

จากคำสั่งข้างต้น จะโดดไปยังคำสั่ง bistwise_2 แต่ไม่ได้เก็บ address ไว้ที่ $ra ดังนั้น register $31 จึงไม่เปลี่ยนแปลง ดังรูป

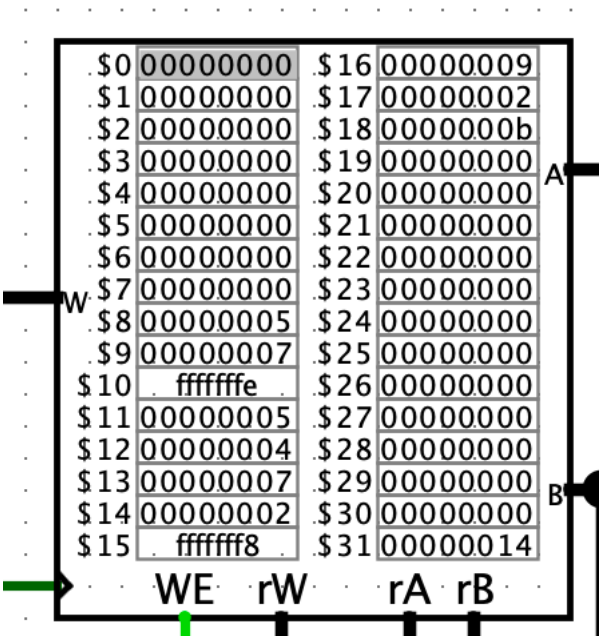


Bistwise_2:

```
andi $t4, $t0, 12
```

อธิบาย :

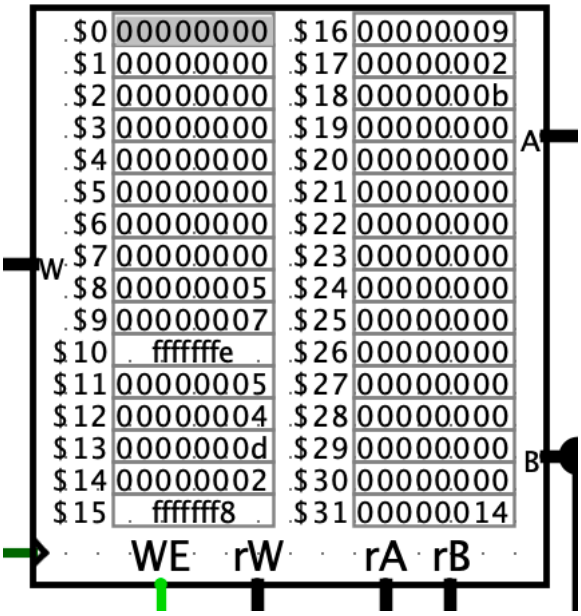
จากคำสั่งข้างต้น จะนำค่าใน $t0 มา AND กับค่าคงที่ 12 และนำค่าไปใส่ไว้ที่ $t4 ดังนั้น Register $12 มีค่าเท่ากับ 0x000004



```
ori $t5, $t0, 13
```

อธิบาย :

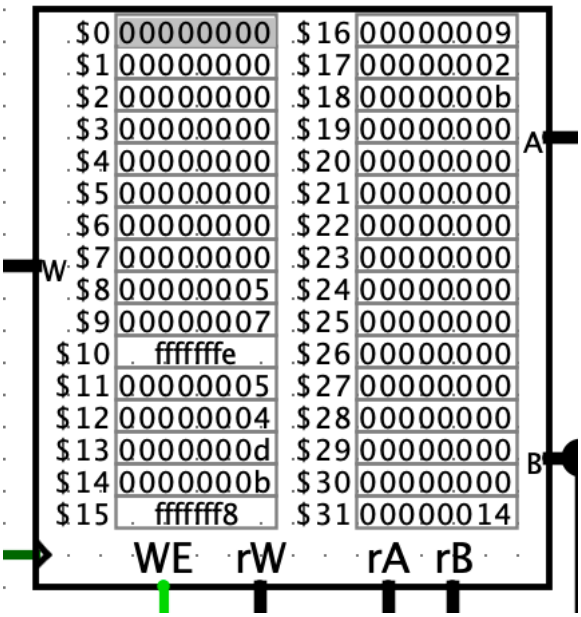
จากคำสั่งข้างต้น จะนำค่าใน $t0 และ ค่าคงที่ 13 มา OR กัน และนำค่าไปใส่ไว้ที่ $t5 ดังนั้น Register $13 มีค่าเท่ากับ 0x00000d




```
xori $t6, $t0, 14
```

อธิบาย :

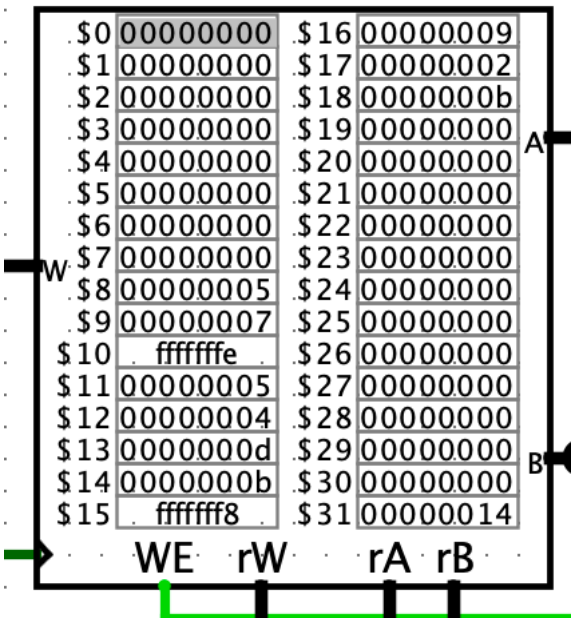
จากคำสั่งข้างต้น จะนำค่าใน $t0 และ ค่าคงที่ 14 มา XOR กัน และนำค่าไปใส่ไว้ที่ $t6 ดังนั้น Register $14 มีค่าเท่ากับ 0x0000000b



```
jr $ra
```

อธิบาย :

จากคำสั่งข้างต้น จะมีการกระโดด ไปที่โปรแกรมทำสุดท้ายก่อนที่จะทำการ jal โดยดู addreass จาก $ra หรือ Register $31 ที่ตอนนี้มีค่า 00000014

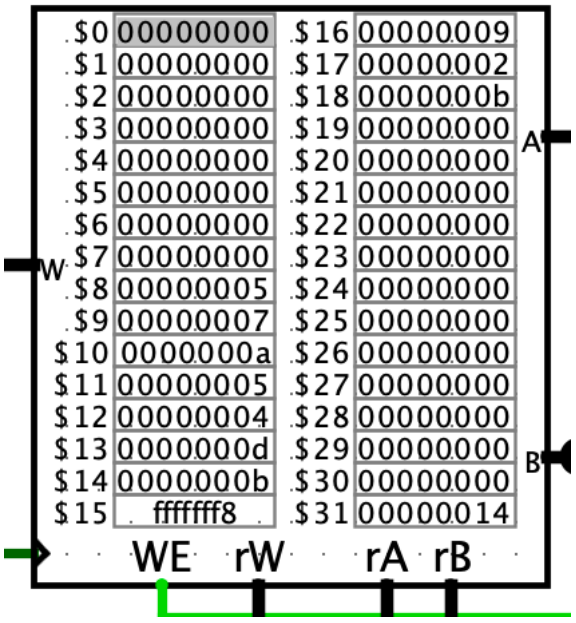


Jal in Main:

```
sll $t2, $t0, 1
```

อธิบาย :

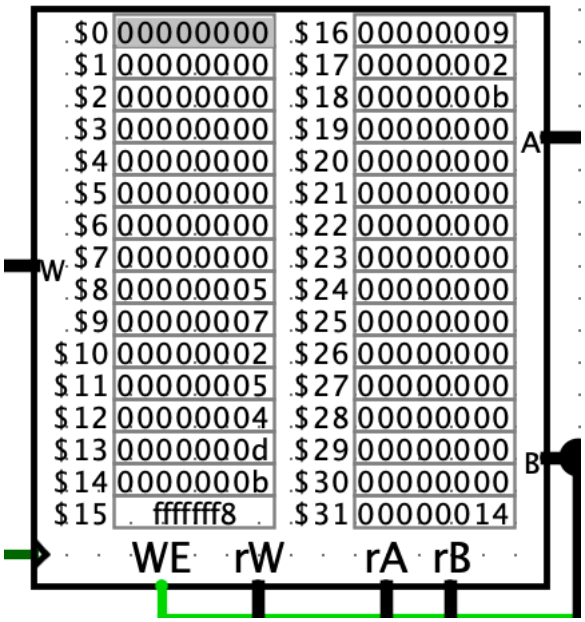
จากคำสั่งข้างต้น จะนำค่าจาก register $t0 ซึ่งมีค่า เท่ากับ 0x000005 มา shift left 1 ครั้ง แล้วนำค่าที่ ได้มาใส่ที่ $t2 ดังนั้นที่ register $10 ถึงมีค่าเท่ากับ 0x00000a



```
srl $t2, $t0, 1
```

อธิบาย

จากคำสั่งข้างต้น จะนำค่าจาก register $t0 ซึ่งมีค่า เท่ากับ 0x000005 มา shift right 1 ครั้ง แล้วนำค่าที่ ได้มาใส่ที่ $t2 ดังนั้นที่ register $10 ถึงมีค่าเท่ากับ 0x000002




```
sra $t2, $t0, 1
```

อธิบาย :

จากคำสั่งข้างต้น จะนำค่าจาก register $t0 ซึ่งมีค่า เท่ากับ 0x000005 มา shift left Arithmetic 1 ครั้ง แล้วนำค่าที่ ได้มาใส่ที่ $t2 ดังนั้นที่ register $10 ถึงมีค่าเท่ากับ 0x000002