**Instruction execution:**

R-type Fetch instruction: Instruction ← MEM[PC]

Fetch operands: data1 ← Reg(Rs), data2 ← Reg(Rt)

Execute operation: ALU\_result ← func(data1, data2)

Write ALU result: Reg(Rd) ← ALU\_result

Next PC address: PC ← PC + 4

I-type Fetch instruction: Instruction ← MEM[PC]

Fetch operands: data1 ← Reg(Rs), data2 ← Extend(imm16)

Execute operation: ALU\_result ← op(data1, data2)

Write ALU result: Reg(Rt) ← ALU\_result

Next PC address: PC ← PC + 4

BEQ Fetch instruction: Instruction ← MEM[PC]

Fetch operands: data1 ← Reg(Rs), data2 ← Reg(Rt)

Equality: zero ← subtract(data1, data2)

Branch: if (zero) **PC ← PC + 4× (1 + sign\_ext(imm16)**

else PC ← PC + 4

LW Fetch instruction: Instruction ← MEM[PC]

Fetch base register: base ← Reg(Rs)

Calculate address: **address ← base + sign\_extend(imm16)**

Read memory: data ← MEM[address]

Write register Rt: Reg(Rt) ← data

Next PC address: PC ← PC + 4

SW Fetch instruction: Instruction ← MEM[PC]

Fetch registers: base ← Reg(Rs), data ← Reg(Rt)

Calculate address: **address ← base + sign\_extend(imm16)**

Write memory: MEM[address] ← data

Next PC address: PC ← PC + 4

Jump Fetch instruction: Instruction ← MEM[PC]

Target PC address: **target ← PC[31:28] || Imm26 || 00**

Jump: PC ← target

|  |  |
| --- | --- |
| **Memory[Address]/register** | **Content** |
| $s1 | 0x20304050 |
| $s2 | 0x10010000 |
| $t1 | 0x10010004 |
| $t2 | 0x10af10fa |
| Memory[0x10010000] | 0xaaaaaaaa |
| Memory[0x10010004] | 0x55433212 |
| Memory[0x1001000c] | 0xcf6789ab |

Based on the above, what are the values for the followings after the instruction execution (instructions are executed individually, not in sequence):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instructions** | **Rs** | **Rt** | **Rd** | **Imm16** | **Address** | **MEM[Address]** |
| ***e.g.***  add $s1,$t2,$s2 | $t2 = 0x10af10fa | $s2 = 0x10010000 | $s1 =  0x20b0 10fa | - | - | - |
| addi $s1,$t2,3 | $t2 = 0x10af10fa | $s1 = 0x10af10fd | - | 3 | - | - |
| lw $s2,8($t1) | $t1 = 0x10010004 | $s2 = 0xcf6789ab | - | 8 | 0x1001000c | MEM[0x1001000c] = 0xcf6789ab |
| sw $s1,4(s2) | $s2 = 0x10010000 | $s1 = 0x20304050 | - | 4 | 0x10010004 | MEM[0x10010004]= 0x20304050 |