Lab2

学号： N/A

姓名： N/A

专业： 计算机科学与技术

Question 1

The main hardware components of the processor are:

1. Program Memory: store and fetch instructions.

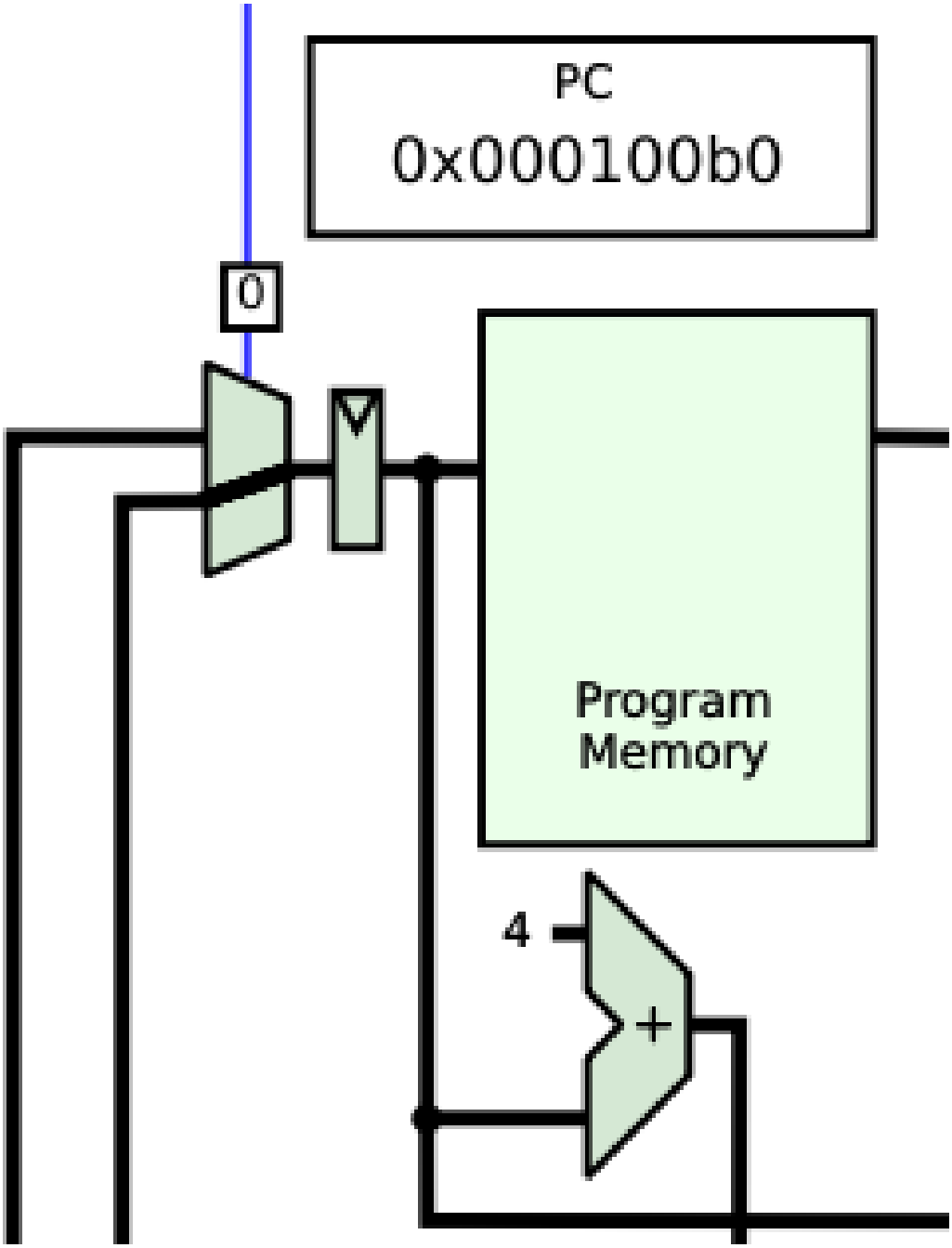

Figure 1: Program Memory

The program memory stores the instructions of the program. The processor fetches the instructions by reading the address from the program counter (PC). PC is updated as PC + 4 after each instruction fetch. If the previous instruction is a branch instruction (e.g., `beq`, `bne`, `jal`, etc.), the PC is updated according to the calculated branch address.

2. Control unit: decode instructions and generate control signals.

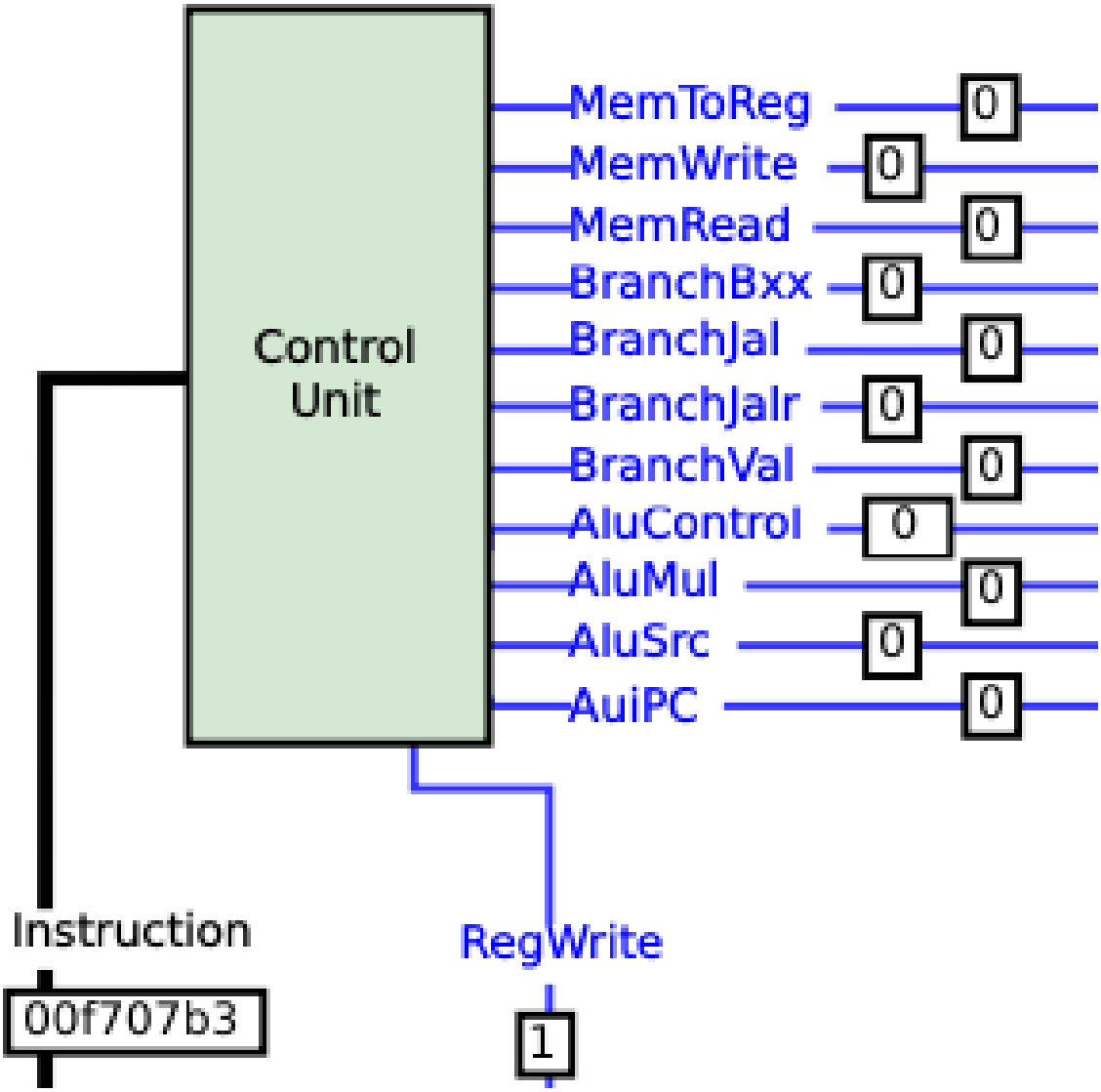

Figure 2: Control Unit

The control unit decodes the instructions and generates control signals for other hardware components. The control signals include:

- RegWrite: write enable signal for the register file.
- MemToReg: mux control signal for selecting the data source for register write: ALU result or memory data.
- MemRead: read enable signal for memory.
- MemWrite: write enable signal for memory.
- BranchBxx: branch control signals (e.g., `beq`, `bne`, etc.).
- BranchJal: jump and link control signal.
- BranchJalr: jump and link register control signal.
- BranchVal: the ALU result for branch calculation, will be used to determine whether to take the branch.
- ALUControl: ALU operation control signal.
- ALUMul: ALU multiplication control signal.
- ALUSrc: ALU source control signal.
- AuiPC: add immediate to PC control signal.

3. Register File: store data and read/write data.

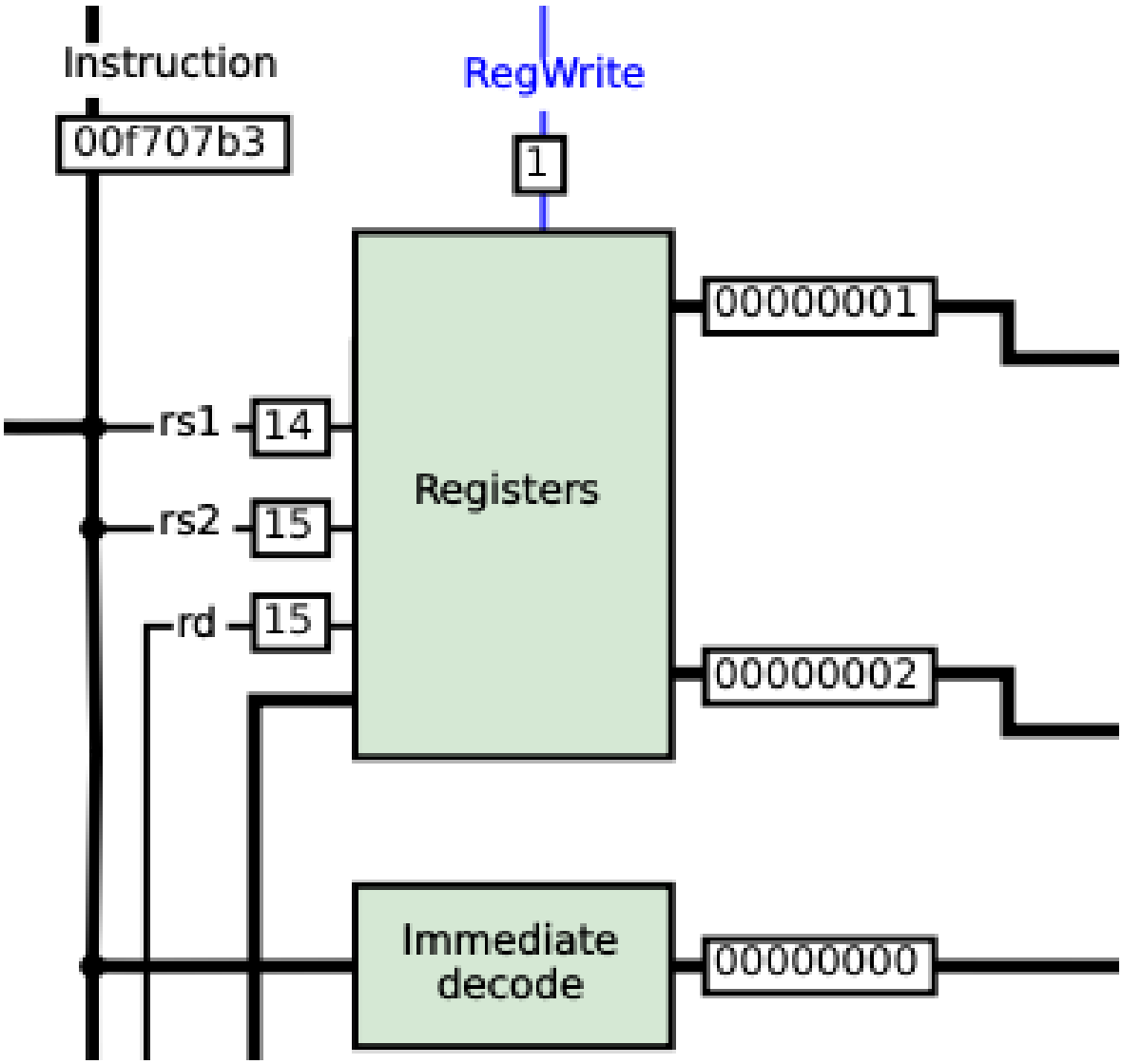

Figure 3: Register File

The register file stores the data and provides read/write access to the registers. The processor has 32 registers, and each register has a unique index. The register file sends the data of rs1 register and rs2 register to the ALU for calculation, and write the rd register with the input data if RegWrite is enabled.

4. ALU: perform arithmetic and logic operations.

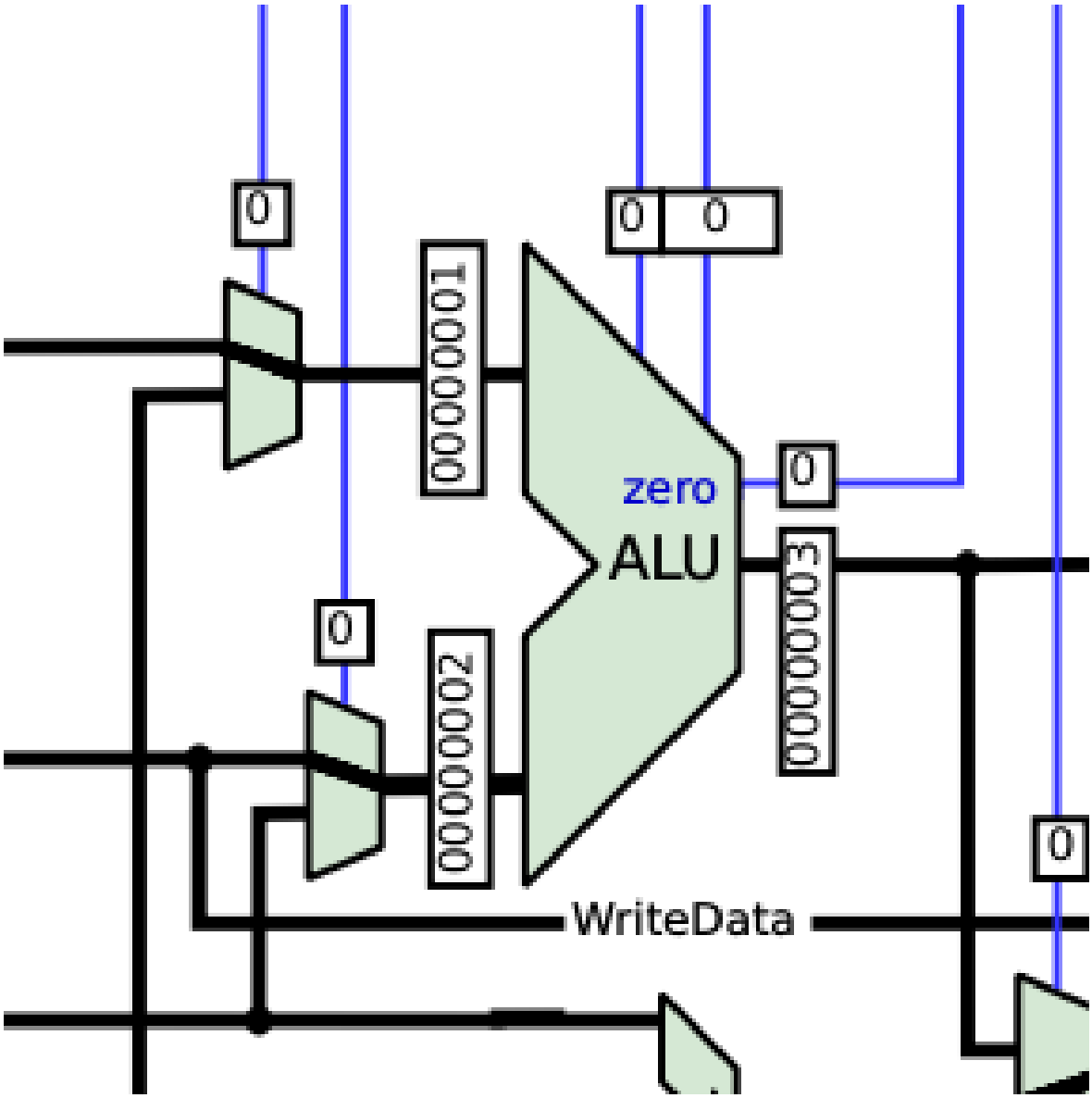

Figure 4: ALU

The ALU performs arithmetic and logic operations on the input data from the register file. The ALU operation is controlled by the ALU control signal generated by the control unit.

- The AuiPC signal selects the source of the first operand of the ALU: the register file or the PC.
- The ALUSrc signal selects the source of the second operand of the ALU: the register file or the immediate value from the instruction.
- The ALUControl and ALUMul signals determine the operation to be performed by the ALU.
- The zero flag output is set if the ALU result is zero, used for branch instructions.

5. Data Memory: store and access data.

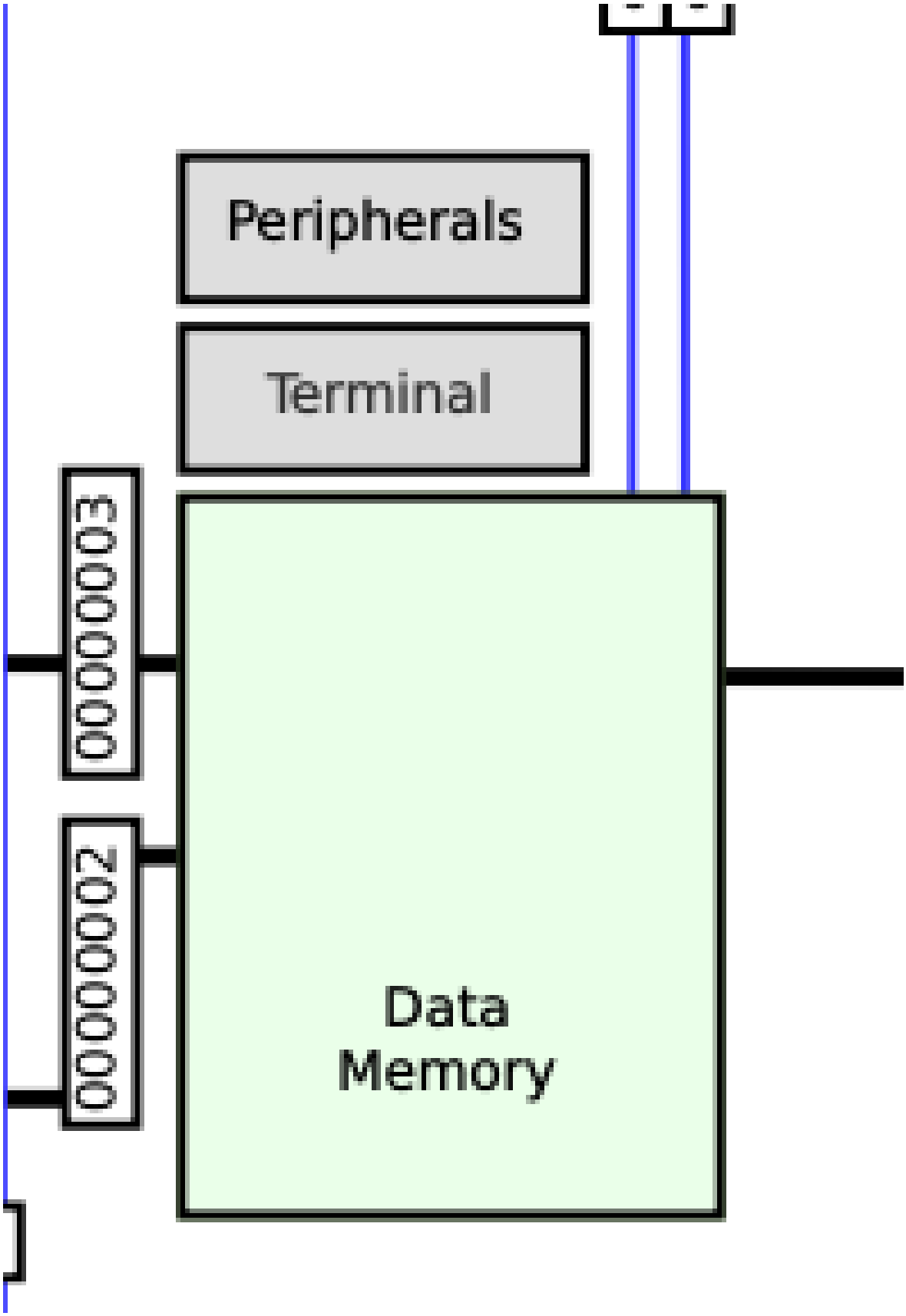

Figure 5: Data Memory

The data memory stores the data and provides read/write access to the data. The input from the ALU is the address for memory access and the input from register file is the data to be written to memory.

- The MemRead signal enables the read operation, and the memory data is sent to the register file if MemToReg is enabled.
- The MemWrite signal enables the write operation.

The processor executes the instructions in the following steps:

1. R-type instruction:
   - Instruction fetch: read the instruction from the program memory.
   - Instruction decode & register read: decode the instruction for rs1, rs2 and rd register numbers and the ALU operation. The register file reads the data from rs1 and rs2 registers.
   - Execution: perform the ALU operation on the data from rs1 and rs2 registers (AuiPC selects rs1, ALUSrc selects rs2), and send result to ALUOut.
   - Memory access: no memory access for R-type instructions.
   - Write back: write the ALU result to the rd register because RegWrite is enabled and MemToReg is disabled.
   - Next PC: PC = PC + 4 because there is no branch instruction.
2. Load instruction:
   - Instruction fetch: read the instruction from the program memory.
   - Instruction decode & register read: decode the instruction for rs1, rd register numbers and the immediate value. The register file reads the data from rs1 register.
   - Execution: perform the addition on the data from rs1 register and the immediate value (AuiPC selects rs1, ALUSrc selects immediate value), and send result to ALUOut.
   - Memory access: memory takes the address from ALUOut and reads the data because MemRead is enabled.
   - Write back: write the memory output to the rd register because RegWrite and MemToReg is enabled.
   - Next PC: PC = PC + 4 because there is no branch instruction.
3. Store instruction:
   - Instruction fetch: read the instruction from the program memory.
   - Instruction decode & register read: decode the instruction for rs1, rs2 register numbers and the immediate value. The register file reads the data from rs1 and rs2 registers.
   - Execution: perform the addition on the data from rs1 register and the immediate value (AuiPC selects rs1, ALUSrc selects immediate value), and send result to ALUOut.
   - Memory access: memory takes the address from ALUOut and the data from rs2 register and writes the data to memory because MemWrite is enabled.
   - Write back: no write back for store instructions.
   - Next PC: PC = PC + 4 because there is no branch instruction.
4. BEQ instruction:
   - Instruction fetch: read the instruction from the program memory.
   - Instruction decode & register read: decode the instruction for rs1, rs2 register numbers and the immediate value. The register file reads the data from rs1 and rs2 registers.

- Execution: perform the subtraction on the data from rs1 and rs2 registers (AuiPC selects rs1, ALUSrc selects rs2), and send zero flag to the control unit.
- Memory access: no memory access for branch instructions.
- Write back: no write back for branch instructions.
- Next PC: PC = PC + 4 if the branch condition is not satisfied, otherwise PC = PC + immediate value << 1.

5. JAL instruction:
   - Instruction fetch: read the instruction from the program memory.
   - Instruction decode & register read: decode the instruction for rd register number and the immediate value. The register file reads the data from rd register.
   - Execution: no execution for jump instructions.
   - Memory access: no memory access for jump instructions.
   - Write back: write PC + 4 to the rd register because RegWrite and jalx is enabled.
   - Next PC: PC = PC + immediate value << 1.
6. JALR instruction:
   - Instruction fetch: read the instruction from the program memory.
   - Instruction decode & register read: decode the instruction for rs1, rd register numbers and the immediate value. The register file reads the data from rs1 register.
   - Execution: perform the addition on the data from rs1 register and the immediate value (AuiPC selects rs1, ALUSrc selects immediate value), and send result to ALUOut.
   - Memory access: no memory access for jump instructions.
   - Write back: write PC + 4 to the rd register because RegWrite and jalrx is enabled.
   - Next PC: PC = ALUOut = rs1 + immediate value << 1.

Question 2

```
7   lw   a4, -20(s0)
```

Load the first parameter `a` from the stack to register `a4`.
Relevant hardware is:
- Program Memory: fetch the instruction.
- Registers: read register `s0` and write register `a4`.
- ALU: calculate the address of the first parameter `s0 - 20`.
- Data Memory: read the first parameter from the stack.

```
8  lw  a5, -24(s0)
```

Load the second parameter `b` from the stack to register `a5`.
Relevant hardware is
- Program Memory: fetch the instruction.
- Registers: read register `s0` and write register `a5`.
- ALU: calculate the address of the second parameter `s0 - 24`.
- Data Memory: read the second parameter from the stack.

```
9  add  a5, a4, a5
```

Add `a` and `b` and store the result in register `a5`.
Relevant hardware is

- Program Memory: fetch the instruction.
- Registers: read register `a4` and `a5`, write register `a5`.
- ALU: calculate the sum of `a` and `b`.

```
10  mv  a0, a5
```

Move the result to register `a0` for return.
Relevant hardware is
- Program Memory: fetch the instruction.
- Registers: read register `a5` and write register `a0`.

Question 3

```
9  add  a5, a4, a5
```

1. Instruction fetch:
   - PC = 0x000100b0
   - Instruction code = 0x00f707b3
2. Instruction decode & register read:
   - Opcode = instruction[6:0] = 0b0110011

输入或输出	信号名称	R型	ld	sd	beq
输入	I[6]	0	0	0	1
	I[5]	1	0	1	1
	I[4]	1	0	0	0
	I[3]	0	0	0	0
	I[2]	0	0	0	0
	I[1]	1	1	1	1
	I[0]	1	1	1	1
输出	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

Figure 6: Opcode decode table

According to the table, the opcode 0b0110011 is `R-type` instruction, and the control signals are:
- RegWrite = 1
- Memory signals:
  - MemToReg = 0
  - MemRead = 0
  - MemWrite = 0
- Branch signals:
  - BranchBxx = 0

    - BranchJal = 0
    - BranchJalr = 0
    - BranchVal = 0
  - ALUOp = 0b10
- rd = instruction[11:7] = 0b01111 = 15
  The destination register is `x15/a5`.
- rs1 = instruction[19:15] = 0b01110 = 14
  The source register 1 is `x14/a4`.
- rs2 = instruction[24:20] = 0b01111 = 15
  The source register 2 is `x15/a5`.
- funct = instruction[31:25, 14:12] = 0b0000000 = 0

ALUOp		funct7字段							funct3字段			操作
ALUOp1	ALUOp0	I[31]	I[30]	I[29]	I[28]	I[27]	I[26]	I[25]	I[14]	I[13]	I[12]	
0	0	X	X	X	X	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	X	X	X	X	0110
1	X	0	0	0	0	0	0	0	0	0	0	0010
1	X	0	1	0	0	0	0	0	0	0	0	0110
1	X	0	0	0	0	0	0	0	1	1	1	0000
1	X	0	0	0	0	0	0	0	1	1	0	0001

Figure 7: ALU control decode table

According to the table, the ALU control signals are:

- ALUControl = 0b0010
  The ALU operation is `add`.
- ALUSrc = 0
- AuiPC = 0

3. Execution:
   - ALU source 1 = rs1 because ALUSrc = 0.
   - ALU source 2 = rs2 because AuiPC = 0.
   - ALU operation = `add` because ALUControl = 0b0010.
   - ALU result = `a4 + a5`.
4. Memory access:
   - No memory access because MemRead = 0 and MemWrite = 0.
5. Write back:
   - Write the ALU result to register `x15/a5` because RegWrite = 1, MemToReg = 0, BranchJal & BranchJalr = 0 and rd = 15.
6. Next PC:
   - PC = PC + 4 = 0x000100b4 because all branch signals are 0.