VIETNAM NATIONLA UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF COMPUTER ENGINEERING
---oOo---

**DIGITAL SYSTEM DESIGN WITH HDL**
**CE213.O12.MTCL.2**

# LAB REPORT 6

**VIETNAMESE NAME : THIẾT KẾ SIMPLE CONTROL UNIT**
**ENGLISH NAME: DESIGING A SIMPLE CONTROL UNIT**

**STUDENT NAME:** Trương Duy Đức
**STUDENT ID:** 21521970
**Lecturer:** Tạ Trí Đức

## Contents

**I. Content of Lab ..... 3**
1.1. Objectives ..... 3
1.2. Practice content ..... 3
**II.THEORETICAL BASIS ..... 4**
1. Control Unit ..... 4
**III.PRACTICAL ..... 5**
1.Control Unit ..... 5
1.1 Control Unit Signal ..... 5
1.2 CODE ..... 7
2. ALU Control ..... 10
2.1 ALU Control Signal ..... 10
2.2 CODE ..... 10
3. Simulation ..... 12
3.1. Transcript ..... 12
3.2. Waveform ..... 13
4. Schematic ..... 13
4.1. Schematic of Control Unit ..... 13
4.2. Schematic of ALU Control ..... 13
**IV. EVALUATION ..... 14**

## Pictures

Figure 1: Data path and Control Unit according to the MIPS architecture. ........................ 3
Figure 2: Signals of the Control Unit and ALU Control. ............................................... 4
Figure 3: signals of the Control Unit and ALU Control. ............................................... 5
Figure 4: Signal table ........................................................................................ 5
Figure 5: ALU State Table .................................................................................. 10
Figure 6: Transcript of Control Unit ..................................................................... 12
Figure 7: Transcript of ALU Control ..................................................................... 12
Figure 8: Waveform of Control Unit ..................................................................... 13
Figure 9: Waveform of ALU Control ..................................................................... 13
Figure 10: Schematic of Control Unit ................................................................... 13
Figure 11: Schematic of ALU Control ................................................................... 13

## I. Content of Lab

### 1.1. Objectives

Using the Verilog HDL language, , design a simple Control Unit.

### 1.2. Practice content

Based on the DATAPATH designed in Lab 4 and relevant theories, students will proceed to design two blocks: Control Unit and ALU Control. These blocks aim to control this DATAPATH as shown in Figure 1 to execute the following instructions using the Verilog HDL language.

- add \$1, \$2, \$3
- lw \$1, 0(\$2)
- sw \$1, 0(\$2)"

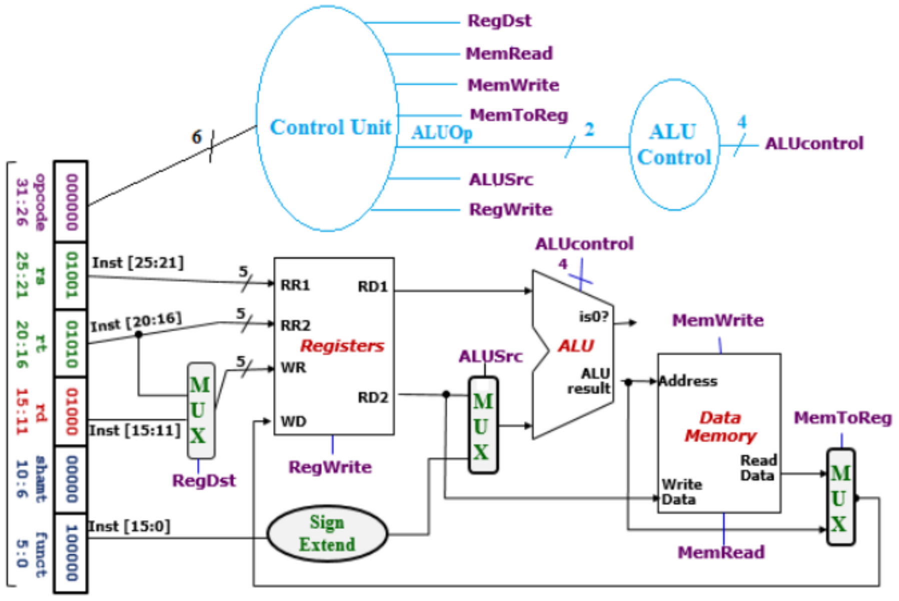


*Figure 1: Data path and Control Unit according to the MIPS architecture.*

## II.THEORETICAL BASIS

### 1. Control Unit

**Control Unit:** The control unit manages the operation of the Data Path and includes the following components:

- **Instruction Decoder:** Decodes instructions to determine specific operations to be performed.
- **Control Signals:** Generates control signals to manage components in the Data Path.
- **PC (Program Counter):** Stores the address of the next instruction in the program.
- **Registers:** Manages access to and updates of registers.
- **ALU Control:** Determines the specific operation of the ALU based on the executing instruction.

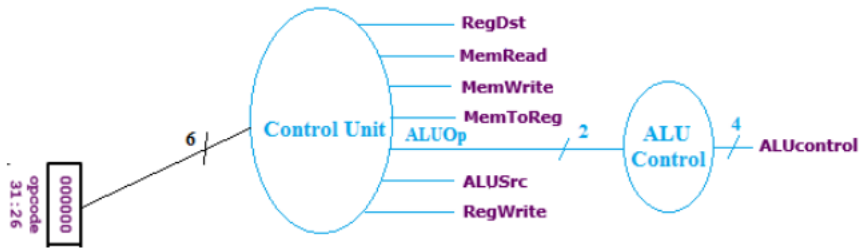


*Figure 2: Signals of the Control Unit and ALU Control.*

## III.PRACTICAL

We will design two blocks, namely the Control Unit and ALU Control with signals as shown in the figure below.

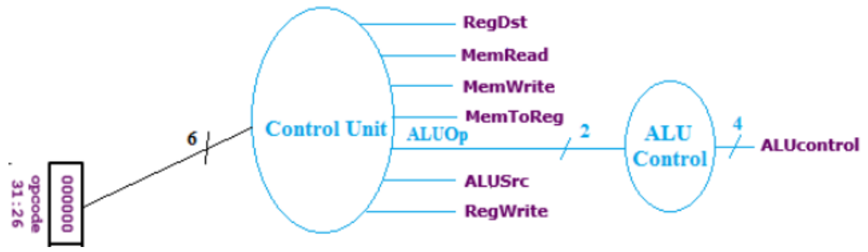


Figure 3: signals of the Control Unit and ALU Control.

The signals will be designed based on the table below:

| Operator | ADD | SUB | AND | OR | LW | SW |
|---|---|---|---|---|---|---|
| Opcode | 1 | 3 | 5 | 7 | 4 | 2 |
| ALUop | 2 | 3 | 0 | 1 | 2 | 2 |
| ALUControl | 5 | 6 | 1 | 3 | 5 | 5 |
| ALU Src | 0 | 0 | 0 | 0 | 1 | 1 |
| REGDST | 1 | 1 | 1 | 1 | 0 | 0 |
| REGWRITE | 1 | 1 | 1 | 1 | 1 | 1 |
| MEMREAD | 0 | 0 | 0 | 0 | 1 | 0 |
| MEMWRITE | 0 | 0 | 0 | 0 | 0 | 1 |
| MEMTOREG | 0 | 0 | 0 | 0 | 1 | 0 |

*Figure 4: Signal table*

### 1.Control Unit

#### 1.1 Control Unit Signal

- Input: Bits 31 to 26 of the instruction (corresponding to the opcode).
- Output signals: REGDST, MEMREAD, MEMWRITE, MEMTOREG, ALUSRC, REGWRITE, each with a length of 1 bit, and ALUop with a length of 2 bits.
- Definitions:
    - **REGDST**: Selects the destination register for the write operation.
        - RegDst = 0: Bits 16 to 20 are selected (rt - for load word instruction).

    - RegDst = 1: Bits 11 to 15 are selected (rd - for other instructions such as add, sub, and, or, slt).

- **RegWrite**: Enables writing to the register file.
    - RegWrite = 0: Register file is read-only (for sw and beq instructions).
    - RegWrite = 1: Register file can perform both read and write operations. The register to be written is specified by the "Write register" input, and the data to be written is taken from the "Write data" input (for other instructions).
- **ALUSrc**: Selects the second input for the ALU.
    - ALUSrc = 0: Second input for ALU comes from the "Read data 2" of the "Registers" block (for add, sub, and, or, slt, beq instructions).
    - ALUSrc = 1: Second input for ALU comes from the output of the "Sign-extend" block (for lw and sw instructions).
- **MemRead**: Enables reading from the Data Memory block.
    - MemRead = 1: Data Memory block reads data. The address to be read is provided by the "Address" input, and the read data is output to the "Read data" (for lw instruction).
    - MemRead = 0: Data Memory block does not read data (for other instructions).
- **MemWrite**: Enables writing to the Data Memory block.
    - MemWrite = 1: Data Memory block writes data. The address to be written is provided by the "Address" input, and the data to be written is taken from the "Write data" input (for sw instruction).
    - MemWrite = 0: Data Memory block does not write data (for other instructions).
- **MemtoReg**: Selects the value to be written to the "Write data" input of the Registers block.
    - MemtoReg = 0: Value comes from ALU (for add, sub, and, or, slt instructions).
    - MemtoReg = 1: Value comes from the Data Memory block (for lw instruction).

## 1.2 CODE

### 1.2.1 CODE of Control Unit

```
module
Control_Unit(OP,REGDST,MEMREAD,MEMWRITE,MEMTOREG,ALUOP,ALUSRC,REGWR
ITE);
input [5:0] OP;
output reg REGDST,MEMREAD,MEMWRITE,MEMTOREG,ALUSRC,REGWRITE;
output reg [1:0] ALUOP;
always @(*)
                case(OP)
                1://add
                begin
                        ALUOP=2'd2;
                        ALUSRC=1'b0;
                        REGDST=1'b1;
                        REGWRITE=1'b1;
                        MEMREAD=1'b0;
                        MEMWRITE=1'b0;
                        MEMTOREG=1'b0;
                end
                3://sub
                begin
                        ALUOP=2'd3;
                        ALUSRC=1'b0;
                        REGDST=1'b1;
                        REGWRITE=1'b1;
                        MEMREAD=1'b0;
                        MEMWRITE=1'b0;
                        MEMTOREG=1'b0;
                end
                5://and
                begin
                        ALUOP=2'd0;
                        ALUSRC=1'b0;
                        REGDST=1'b1;
                        REGWRITE=1'b1;
                        MEMREAD=1'b0;
                        MEMWRITE=1'b0;
                        MEMTOREG=1'b0;
                end
                7://or
                begin
                        ALUOP=2'd1;
                        ALUSRC=1'b0;
                        REGDST=1'b1;
                        REGWRITE=1'b1;
                        MEMREAD=1'b0;
                        MEMWRITE=1'b0;
                        MEMTOREG=1'b0;
                end
                4://lw
                begin
                        ALUOP=2'd2;
                        ALUSRC=1'b1;
                        REGDST=1'b0;
                        REGWRITE=1'b1;
                        MEMREAD=1'b1;
                        MEMWRITE=1'b0;
                        MEMTOREG=1'b1;
                end
                2://sw
                begin
                        ALUOP=2'd2;
                        ALUSRC=1'b1;
                        REGDST=1'b0;
                        REGWRITE=1'b1;
                        MEMREAD=1'b0;
                        MEMWRITE=1'b1;
                        MEMTOREG=1'b0;
                end
                default:
                begin
                        ALUOP=2'dx;
                        ALUSRC=1'bx;
                        REGDST=1'bx;
                        REGWRITE=1'bx;
                        MEMREAD=1'bx;
                        MEMWRITE=1'bx;
                        MEMTOREG=1'bx;
                end
        endcase
endmodule
```

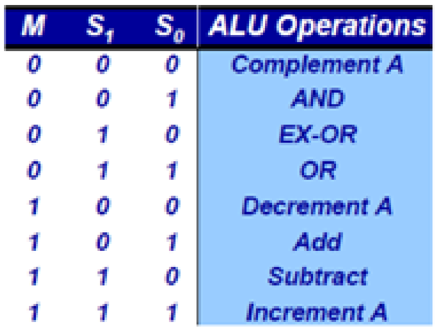
### 1.2.2 CODE of Control Unit Testbench

```
`timescale 1ns / 100ps
module tb_Control_Unit;
reg [5:0]OP;
wire REGDST,MEMREAD,MEMWRITE,MEMTOREG,ALUSRC,REGWRITE;
wire [1:0] ALUOP;
reg [31:0] count=1;
Control_Unit
A(.OP(OP),.REGDST(REGDST),.MEMREAD(MEMREAD),.MEMWRITE(MEMWRITE),.
MEMTOREG(MEMTOREG),.ALUOP(ALUOP),.ALUSRC(ALUSRC),.REGWRITE(REGW
RITE));
initial begin
        #0
        OP=6'b000000;
        #100 $stop;
end
always #10 begin
#1 $display("CASE %d---OPCODE:%b",count,OP);
#1 $display("REGDST:%b, MEMREAD:%b, MEMWRITE:%b, MEMTOREG:%b,
ALUSRC:%b,
REGWRITE:%b",REGDST,MEMREAD,MEMWRITE,MEMTOREG,ALUSRC,REGWRITE
);
$display("ALUOP:%b",ALUOP);count=count+1;OP=OP+1;
end

endmodule
```

## 2. ALU Control

### 2.1 ALU Control Signal

- Input: ALUop.
- Output: ALUcontrol with a length of 4 bits.
- Definitions:
    - ALUcontrol is the signal to select the operation to be performed in the ALU (block designed in Lab 4)



| $M$ | $S_1$ | $S_0$ | ALU Operations |
|---|---|---|---|
| 0 | 0 | 0 | Complement A |
| 0 | 0 | 1 | AND |
| 0 | 1 | 0 | EX-OR |
| 0 | 1 | 1 | OR |
| 1 | 0 | 0 | Decrement A |
| 1 | 0 | 1 | Add |
| 1 | 1 | 0 | Subtract |
| 1 | 1 | 1 | Increment A |

*Figure 5: ALU State Table*

### 2.2 CODE

#### 2.2.1 CODE ALU control

```
module  Alu_control(ALUOP,ALUcontrol);

input [1:0] ALUOP;
output reg[3:0] ALUcontrol;

always @(*)
        case(ALUOP)
                0: ALUcontrol=4'd1;//AND
                1: ALUcontrol=4'd3;//OR
                2: ALUcontrol=4'd5;//ADD
                3: ALUcontrol=4'd6;//SUB
        endcase
endmodule
```

### 2.2.2 CODE ALU control Testbecnh

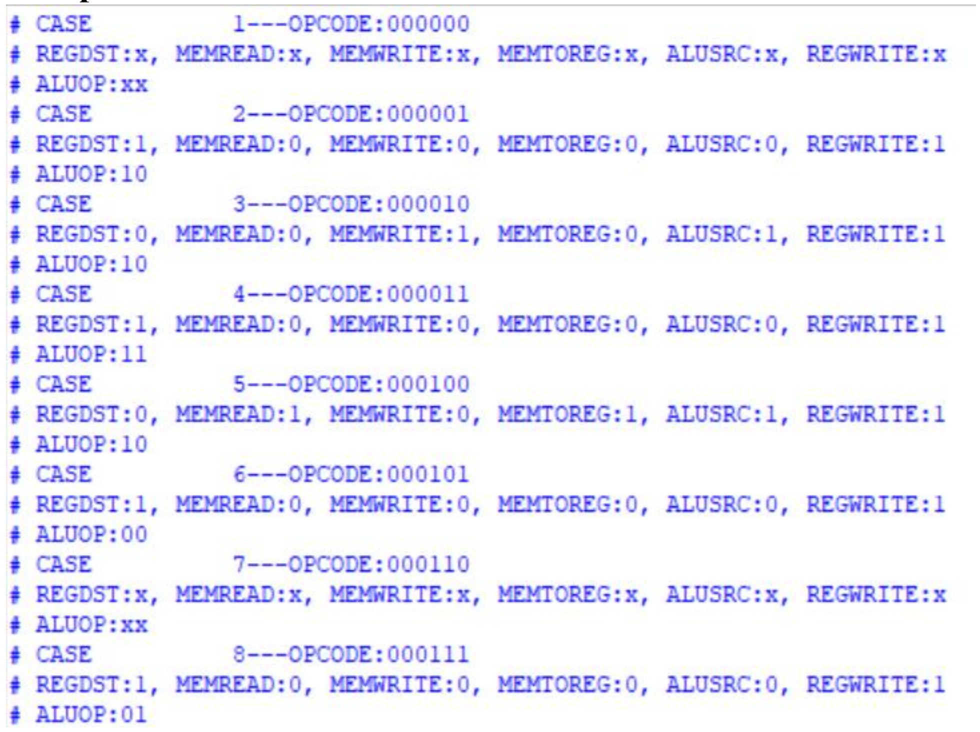
```
`timescale 1ns / 100ps
module tb_Alu_control;
reg [1:0] ALUOP;
wire [3:0] ALUcontrol;
reg [1:0] count;
Alu_control a(.ALUOP(ALUOP),.ALUcontrol(ALUcontrol));
initial begin
        #0
        ALUOP=2'b00;
        #60 $stop;
end
always #10 begin
#1 $display("CASE %d---ALUOP:%b",count,ALUOP);count=count+1;ALUOP=ALUOP+1;
        $display("ALU_control: %b",ALU_control);
end

endmodule
```

## 3. Simulation

Check to see if all cases are correct. Based on Figure 4, we will input data and observe the output, then all cases are correct

### 3.1. Transcript

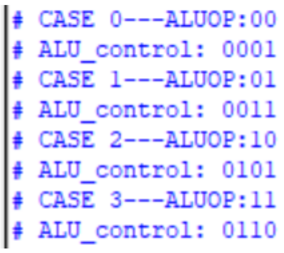
#### 3.1.1. Transcript of Control Unit



```
# CASE          1---OPCODE:000000
# REGDST:x, MEMREAD:x, MEMWRITE:x, MEMTOREG:x, ALUSRC:x, REGWRITE:x
# ALUOP:xx
# CASE          2---OPCODE:000001
# REGDST:1, MEMREAD:0, MEMWRITE:0, MEMTOREG:0, ALUSRC:0, REGWRITE:1
# ALUOP:10
# CASE          3---OPCODE:000010
# REGDST:0, MEMREAD:0, MEMWRITE:1, MEMTOREG:0, ALUSRC:1, REGWRITE:1
# ALUOP:10
# CASE          4---OPCODE:000011
# REGDST:1, MEMREAD:0, MEMWRITE:0, MEMTOREG:0, ALUSRC:0, REGWRITE:1
# ALUOP:11
# CASE          5---OPCODE:000100
# REGDST:0, MEMREAD:1, MEMWRITE:0, MEMTOREG:1, ALUSRC:1, REGWRITE:1
# ALUOP:10
# CASE          6---OPCODE:000101
# REGDST:1, MEMREAD:0, MEMWRITE:0, MEMTOREG:0, ALUSRC:0, REGWRITE:1
# ALUOP:00
# CASE          7---OPCODE:000110
# REGDST:x, MEMREAD:x, MEMWRITE:x, MEMTOREG:x, ALUSRC:x, REGWRITE:x
# ALUOP:xx
# CASE          8---OPCODE:000111
# REGDST:1, MEMREAD:0, MEMWRITE:0, MEMTOREG:0, ALUSRC:0, REGWRITE:1
# ALUOP:01
```

*Figure 6: Transcript of Control Unit*

#### 3.1.2. Transcript of ALU Control



```
# CASE 0---ALUOP:00
# ALU_control: 0001
# CASE 1---ALUOP:01
# ALU_control: 0011
# CASE 2---ALUOP:10
# ALU_control: 0101
# CASE 3---ALUOP:11
# ALU_control: 0110
```

*Figure 7: Transcript of ALU Control*

## 3.2. Waveform

### 3.2.1. Waveform of Control Unit

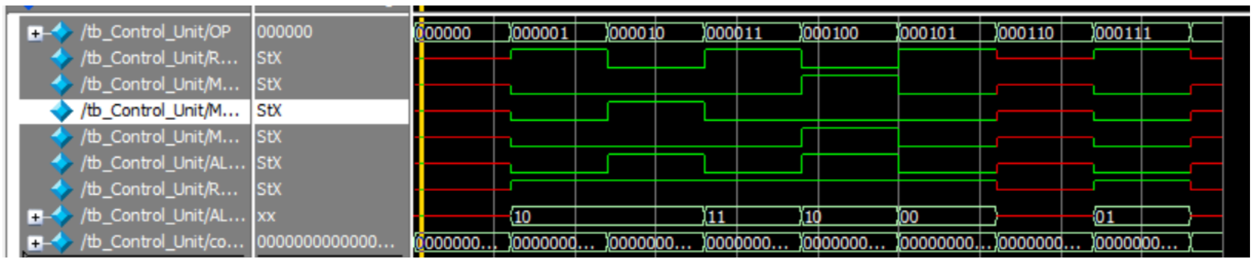


Figure 8: Waveform of Control Unit

### 3.2.2. Waveform of ALU Control

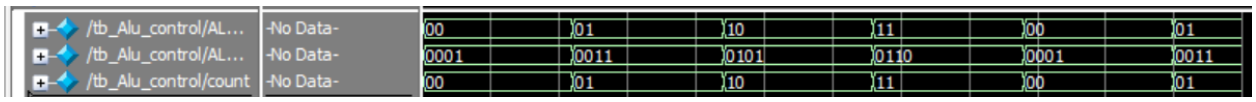


Figure 9: Waveform of ALU Control

## 4. Schematic

### 4.1. Schematic of Control Unit

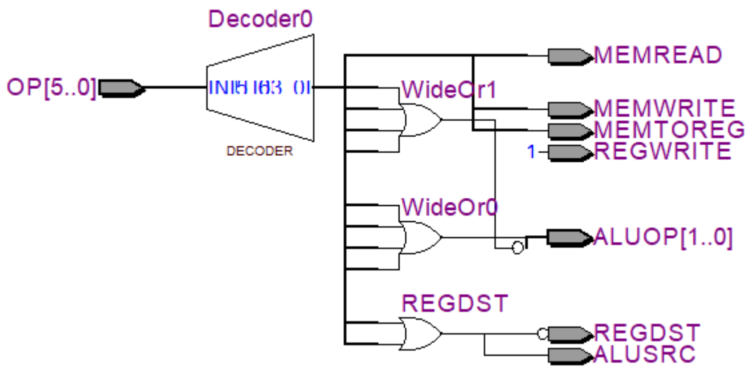


Figure 10: Schematic of Control Unit

### 4.2. Schematic of ALU Control

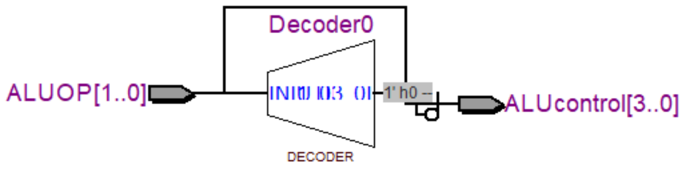


Figure 11: Schematic of ALU Control

## IV. EVALUATION

No errors were found, and the circuit runs as intended.