CSE417

MICROPROCESSOR

HOMEWORK 1

Sibel Aymış 150112075

Sema Çevik 150112077

CONTROL UNIT

We implemented control unit by using ROM.We have 20 bit values in ROM and these values addresses are determined according to operation alucodes.In our circuit:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Retmux | Mux2 | Fetch | Pcld | Pc++ | Zfld | İrld | Armux1 | Armux0 | Mux1 | Mux0 | Memwt | Sp-- | Sp++ | Regld | nextAdd  0-4 |  |
| 0 |  |  | 1 |  | 1 |  | 1 |  |  |  |  |  |  |  |  | 0 | fetch |
| 2 |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  | 1 | 0 | ldi |
| 4 |  |  |  |  |  |  |  |  | 1 |  |  |  |  |  | 1 | 0 | ld |
| 6 |  |  |  |  |  |  |  |  | 1 |  | 1 | 1 |  |  |  | 0 | st |
| 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 0 | jz |
| A |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  | 0 | jmp |
| C |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 0 |  |
| E |  |  |  |  |  | 1 |  |  |  | 1 | 1 |  |  |  | 1 | 0 | alu |
| 10 |  |  |  |  |  |  | 1 | 1 |  |  | 1 | 1 | 1 |  | 1 | 0 | push |
| 12 |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  | C | pop |
| 14 |  | 1 |  | 1 |  |  |  | 1 |  |  |  |  | 1 |  | 1 | 0 | call |
| 16 |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  | D | ret |
| 18 |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  | 0 | Pop1 |
| 1A | 1 |  |  | 1 |  |  |  | 1 |  |  |  |  |  |  |  | 0 | Ret1 |

0 2a000 00000 FETCH

2 08020 00000 LDI

4 00820 00000 LD

6 00b00 00000 ST

8 00000 00000

A 10000 00000 JMP

C 00000 00000

E 04620 00000 ALU

10 01380 00000 PUSH

12 00053 01020 POP

14 51180 00000 CALL

16 00057 91000 RETURN

We considered this control unit by considering that each instruction is completed in two clock cycle.

LDI

1006 0010

Ldi 6 A1

ldi 6 A1 meaning that put address of the A1 immediate value to sixth registerIn first cycle 1006 hexadecimal value is read and it went to instruction register. Also we put one 0 to end of first four bit of instruction(0001) .00010 went to control unit showing that 2nd adress of ROM.This happened because of our control unit design,opcodes signals stays at the address of two times alucode.

Ldi 6 A1 //sixth register has A1 s address

Ld 6 6 //Sixth registers adress is used in the memory and in that place there is A1 value.This value will be loaded into sixth register.

LD

ld 5 5 fifth registers content points a memory location,value in that location is put into fifth register.

ST

st 2 3 meaning that store content of third register to the memory whose address is pointed by content of the second register.

ALU

In all ALU operations mux is 11 saying that alu result will enter to the register file as register input.To write alu output into destination register regload is 1.And zero flag will be changed according to alu operation so zfld is also 1.

JUMP

In jump instruction,only pcload is 1.Everything else is 0.Instruction registers hold the address of the label.This address enters to adder and from there it enters to PC.Pcload is 1 so this value is written into PC.ARMUX is 0 so that next instructions address is this address.

JZ

In this mux JZ is handled here 0 th bit of the control signal is ZF first bit is isJZ. If isJZ is not 1 cpu uses opcode,if isJZ 1 but ZF is 0 then control unit fetches to next instruction without jump.If they are both 1 jump happens as we explained above.

CALL

Call is one kind of jump.Only difference is next address of the call instruction is stored into stack.In call instruction armux = 10,mux=100,memwrite=1,pcload=1,sp--=1.In this instruction content of the stack pointer which is 7 th register of the register file enters to address input of memory via armux=10.Stack pointer always points to a empty field.This field is where we will put the next address of the currently executed instruction address.Mux selects bit is 4.This means value inside pc(currently executed instructions address + 1) goes to data input for memory.So that we are able to write next addresses value into the top of the stack.And sp-- is 1 so stack pointer will again show empty field as it should be.One thing remains: which is how to jump.Lets explain this.Instruction registers datas belong to currently executed instructions datas and does not affected by memory address change(instruction register property).So instruction register holds value of the label.It is again pc relative so that it enters to adder.retmux is 0 so that this sum enters to PC.Pcload is 1 so that next instructions address will be result of this sum.So that jump will happen.

RETURN

In return we first increment stack pointer to get to the top of the stack.Then in the second cycle we load that value at the top of the stack to PC.So that we return into address which we assume that we have stored in top of the stack previously.

In first cyle sp++ is 1 only.In the second cycle armux=2,mux=0,retmux=1,pcload=1.This instruction resembles pop but it is different.regload,pcload and retmux has different values in these two instruction.Armux is 2 so that address of the memory comes from stack pointer.Output of the memory is the top of the stack.This enters to mux which is controlled by retmux.Retmux is 1 so this value passes through PCs input.This value is written to the PC because pcload is 1.So next instructions address will be the value stored in the top of the stack.

POP

Pop used for putting the value at the top of the stack into a specified register.In first cycle sp++ signal is 1 only. Stack pointer points to empty field normally,to get the value at the top of the stack we increment sp by one so that now it points to the top of the stack instead of empty field.Then in the second cycle:Armux is 2,mux is 0 and regload is 1.When armux is 2,memory shows the address shown by stack pointer.We have incremented sp in previous cycle.So output of the memory in second cycle of the pop instruction equals to the value at the top of the stack.It enters to mux below and mux selects are 0 so that it enters to reginput input of the registerfile.Registerdestination input comes from instruction register as it always.Regload is 1 so that the value coming from memory is written to destinaton register.

PUSH

In this instruction we put the content of the given register to the top of the stack.In this instruction armux=10,mux is 001,memwrite is 1 and sp - - is 1.Armux is 10 so that content of 7th register(memory pointed by stack pointer) enters address input of memory.Mux is 1 right muxes output passes the mux enters to data input of memory.memwrite is 1 so that value in the data input is written into address which is stack pointers memory.And sp – is 1 so that stack pointer points to empty field as it should be.