CPU DESIGN USING LOGISM

J.Himani

Introduction

16 bit CPU designed using RISC architecture.

The CPU has a 16-bit address bus and a 16-bit data bus.
There are 8 general purpose 16-bit registers, an ALU which can do 9 different operations, an instruction register and an immediate register to hold immediate values.
Each simple instruction is 1-word (16 bits) long with these fields:

- 7-bit opcode (MSBs)
- 3-bit t-register (usually the 2nd operand to the ALU)
- 3-bit s-register (the 1st operand to the ALU)
- 3-bit d-register (LSBs, usually the destination of the ALU operation)

Extended instructions are two words long where the first word has the same format as above and is followed by a 16-bit immediate/literal value.

The ALU

The ALU takes two 16-bit data inputs and produces a 16-bit data output as well as zero, negative and carry flags. The *aluop* control line specifies which of the 9 available operations to perform.

The 9 available operations are:

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. And
6. Or
7. Xor
8. Nor
9. Nand

The Register File

The register file contains eight 16-bit general purpose registers.
The *dsel*, *ssel* and *tsel* control lines select which of the eight registers become the d-register, s-register and t-register.
The d-register is generally used as the destination register for ALU operations.
The s- and t-registers are used as ALU sources.
The *regval* input to the ALU is written to a register when one of the *dwrite*, *swrite* or *twrite* lines and enabled.
The source of *regval* is determined by the multiplexer and is controlled by the *regsrc* line.
The options are: from the data bus, from the immediate register, from the ALU output, or from the s-register.

CODES FOR REGISTERS:

REGISTER	CODE
R0	000
R1	001
R2	010
R3	011
R4	100
R5	101
R6	110
R7	111

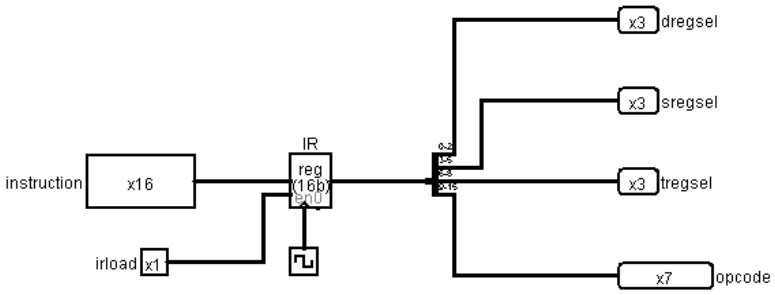
The Immediate Register

It holds the immediate second word from the 2-word instructions.
It is loaded from the data bus when the *imload* control line is enabled.

The Instruction Register

The 16-bit IR is loaded from the data bus when the *irload* control line is enabled.
It just splits out the 16-bits into the 7-bit *opcode* and 3-bit *dsel*, *ssel* & *tsel* control lines.
It is a single register with a bit-splitter on the right.

Instruction Register



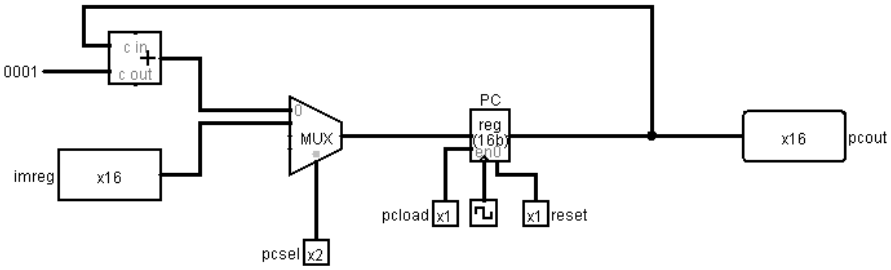
The Program Counter

The PC is a 16-bit register which is loaded when the *pcload* control line is enabled.
Its new value is controlled by the *pcsel* control line, and allows these inputs:

- PC+1, i.e. increment the PC
- immediate register, used to jump the PC to an absolute address.

PC's new value is chosen by a multiplexer which receives PC+1 using an adder, the immediate register.

Program Counter



The Data Bus

Data normally flows from RAM onto the 16-bit data bus and this goes into the immediate register, the IR, and into the multiplexer before the register file.
However, when the *datawrite* control line is enabled, it allows writes out to RAM. There is a controlled buffer which lets the data out onto the bus from the CPU. The multiplexer controlled by the *datasel* line chooses what to write on the data bus i.e the d-register, the t-register or the ALU output.

The Address Bus

The 16-bit address bus gets its value from one of two inputs controlled by the *addrsel* control line: the PC, the immediate register. This allows such addressing modes as:

- PC: fetch the next instruction
- immediate: fetch from a fixed memory location

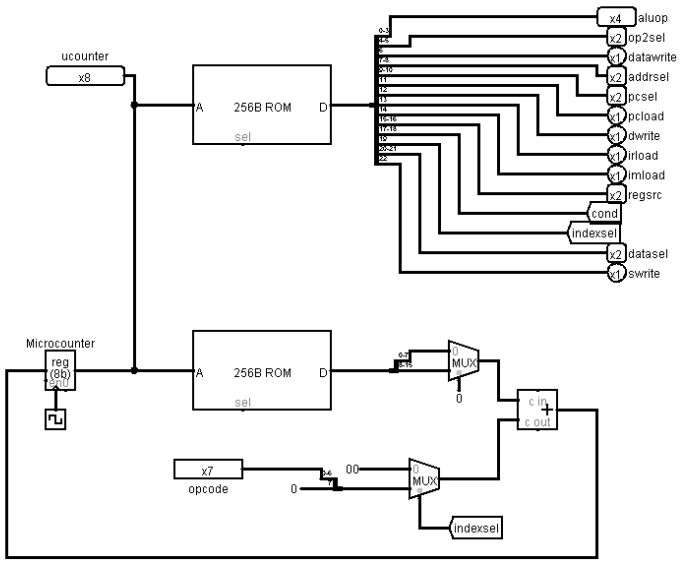
The Microprogram Logic

Each machine-code instruction is interpreted by the microcode logic as several microinstructions. Each microinstruction enables certain control lines which do things as update a register's value, command the ALU to perform a certain operation, fetch a value from memory etc. After each microinstruction, the microcode logic has to determine which next microinstruction to perform.

The microcode unit takes as inputs the current *opcode*. It then outputs all the control lines which control the components and busses: *aluop*, *op2sel*, *datawrite*, *addrsel*, *pcsel*, *pcload*, *dwrite*, *irload*, *imload*, *regsrc, datasel* and *swrite*.

Internally, the microcode logic is implemented using two ROMs and a micro counter. Each ROM has 256 rows where each row stores a single microinstruction. The control ROM is 32-bits wide, and on each row the bits enable or disable all the CPU control lines for this microinstruction. At present, only 23 out of the 32 bits are used.

Control Unit



INSTRUCTION FORMAT:

1 WORD=16 BITS (7+3+3+3=16)

OPCODE (7 bits)	TARGET REGISTER (Rt-3 bits)	SOURCE REGISTER (Rs-3 bits)	DESTINATION REG (Rd-3 bits)

OPCODES:

Instruction	Opcode
Load	0111111
Store	1000001
Add	0000000
Sub	0000001
Mul	0000010
Div	0000011
Rem	0000100
And	0000101
Or	0000110
Xor	0000111
Nand	0001000
Nor	0001001
Move	1000110

CONTROL SIGNALS FOR INSTRUCTIONS:

Fetch:
addrsel=pc, irload=1, pcload=1 , pcsel=pc

ALU operations on Rd,Rs,Rt:
aluop=add , op2sel=treg , dwrite=1, regsrc=aluout
Similarily for other ALU operations also.

Load Rd, immed:
addrsel=pc , dwrite=1, regsrc=databus

Store Rd into address from immed:
addrsel=pc, imload=1,addrsel=immed , datawrite=1, datasel=dreg

Move Rd,Rs:
addrsel=pc, dwrite=1, regsrc=sreg

SAMPLE PROGRAM:

```
load R0,6                   // loads 6 into R0
load R1,2                   //loads 2  into R1
sub R0,R0,R1                // R0 <- R0 – R1     R0=6-2=4
mul R0,R0,R1                // R0 <- R0*R1     R0=4*2=8
add R0,R0,R1                // R0 <- R0 + R1    R0=8+2=10
store R0, f                 // stores value of R0 in address 000f in the RAM.
```

HEXADCIMAL CODE:

The following is the hexadecimal code to the above code to be given as input in the RAM in the main of CPU.circ.

RAM ADDRESS	HEX CODE		
0000:	7e00	load R0,6	// loads 6 into R0
0001:	0006	Value to be loaded	//6
0002:	7e01	load R1,2	// loads 2 into R1
0003:	0002	Value to be loaded	//2
0004:	0240	sub R0,R0,R1	// R0 <- R0 – R1 R0=6-2=4
0005:	0440	mul R0,R0,R1	// R0 <- R0*R1 R0=4*2=8
0006:	0040	add R0,R0,R1	// R0 <- R0 + R1 R0=8+2=10
0007:	8200	store R0, a	// stores value of R0 in address 000f in the RAM.
0008:	000f		//Address where the value of register has to be stored