Computer Architecture Project (BU2020 Processor)

Project title: Design and implementation of a RISC processor
Department: Computer Engineering Department, Bogazici University

Project description

As a partial fulfilment of Computer Architecture course, the students should design a 16 bit-RISC processor), the so called BU2020 processor, using a hardware description language (VHDL or Verilog). BU2020 Processor is a 16 bit-RISC processor and it has 5 set of instructions.

- Arithmetic and Logical instructions
- Immediate instructions
- Transfer instructions
- Memory access instructions
- Jump instructions

The project includes three parts: 1) Datapath Design, 2) control path design, 3) design integration and verification. The required contents, documents, and the deadlines for each part will be announced.

1. Register sets

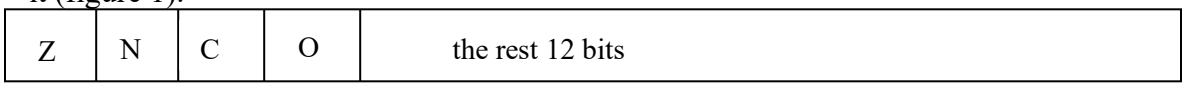
BU2020 has 11 registers of 16-bit length. Table 1 shows BU2020's registers and their description.

Table 1: BU2020 processor registers and their description

Data Registers	D0 ~ D3	4 fast registers to perform arithmetic and logic
Address Registers	A0 ~ A2	3 register to access the memory
Base Address	BA	Base address
Program counter	PC	Program counter
Status register	SR	Status register
Zero register	Zero	Always equal to the Zero

2. Status register

Status register (flag register) like as other registers is a 16-bit register, but you just need the first four bits of it (figure 1).



Z = 1 if result =0;
N = 1 if result<0;
C = 1 if result has carry;
O = 1 if result has overflow;

Figure 1: Status register

Table 2: BU2020 processor assembly language instruction set

Category	Instruction	Example	Meaning	comments
Arithmetic	add	Add d0, d1,d2	d0= d1 + d2	Two operands; overflow detected (opcode =0000)
	add	Add d0,d1, 25	d0 = d1 + memory(BA+25*2)	Two operands; overflow detected (opcode = 0001)
	subtract	Sub d0, d1,d2	d0 = d1 – d2	Two operands; overflow detected (opcode = 0010)
	Add immediate	Addi d0,d1, 24	d0 = d1 + 50	+ constant; overflow detected (opcode = 0011)
	Multiply	Mul d0, d1,d2	d0 = d1 × d2	32 bit signed product. 16 bits LSB part in d0 (opcode = 0100)
Logical	And	And d0, d1,d2	d0 = d1 & d2	Two reg. operand; bitwise logical and (opcode = 0101)
	Shift left logical	Sll d0, 10	d0 = d0<<10	Shift left by constant (opcode = 0110)
Data Transfer	Load word	Lw d0, 7	d0 = memory(BA + 14)	Word from memory to register (14 = 7×2) (opcode = 0111)
	Load indirect word	Lwi d0,7	d0 = mem(mem(BA+14))	Word from memory of memory is moved to reg (opcode = 1000)
	Store word	Sw d0, 10	Memory(BA+ 20) = d0	Word from register to memory (20= 10×2) (opcode = 1001)
	Store indirect word	Swi d0,10	Mem(Mem(BA + 20)) = d0	Reg is store in memory of memory location (opcode = 1010)
	Clear reg or mem	CLR d0	d0 = 0	One operand; clear reg or clear memory (opcode = 1011)
	Move immediate	Mov BA, 50	50 → BA	Move immediate to BA reg (opcode = 1100)

Compare and conditional branch	compare	CMP d0, d1	If (d0-d1=0)
			Z flag = 1
			Elsif (d0 – d1<0)
			N flag = 1
			Else
			Z flag and N flag are zero
				It does not change content of operands. CMP instruction change content of status register (flags) (opcode = 1101)

	Branch on not equal	Bne 25	PC = PC + 2 + 50	Bne checks Z flag and if Z flag is zero, it go to location (25 × 2); PC relative (opcode = 1110)
Unconditional Jump	Simple jump	Jmp 2500	Go to 5000	Jump to target address (2500 × 2) (opcode = 1111)

3. Assembly language of BU2020 processor

As it has been mentioned above BU2020 has five set of instructions. Table 2 shows BU2020 ISA and their description as well as an example for each of them.

4. Memory access in BU2020

As any word in BU2020 has 2 bytes (16 bits), words must always start at addresses that are multiply of 2. This requirement is called an alignment restriction, and many architectures have it. This type of addressing is known as word addressing. Figure 3 shows BU2020 memory addresses. For example the address of third word is 8.

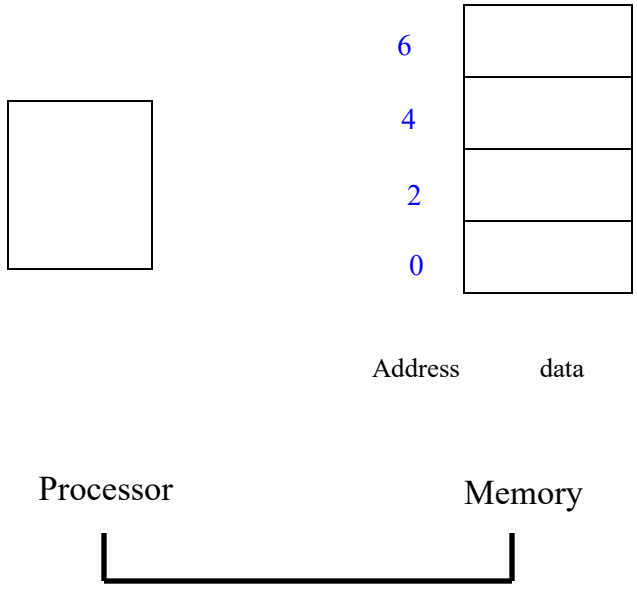


Figure 3: word addressing in BU2020 processor

5. Types of memory in BU2020

The BU2020 processor has two separate memories for instruction and data memory. Instruction memory has 1 KB and data memory has 3KB.

6. Machine code in BU2020 processor

BU2020 has two formats (R-format and I format) for machine code of instructions as follows

6.1. R-format

The R-format (Register format) is shown below (figure 4):

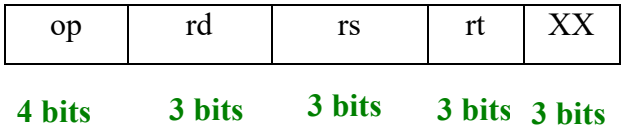


Figure 4: R-format

- ❖ op: Basic operation of the instruction, traditionally called *opcode*.
- ❖ rd: The destination register
- ❖ rs: The source register 1
- ❖ rt: The source register 2
- ❖ XX: don't care

- ✓ The instruction of "CMP d0, d1" locates in this category. But the rd register becomes neutral and the contents of the rd register is don't care here. CMP indirectly updates the status register. So, you don't need represent the status register in the format of instructions.
- ✓ The "CLR d0" is a R-format instruction. d0 is the destination register (i.e., rd) and rs and rt are neutral here and their contents are don't cares.

6.2 I-format

The I-format has two representations as follows (figure 5):

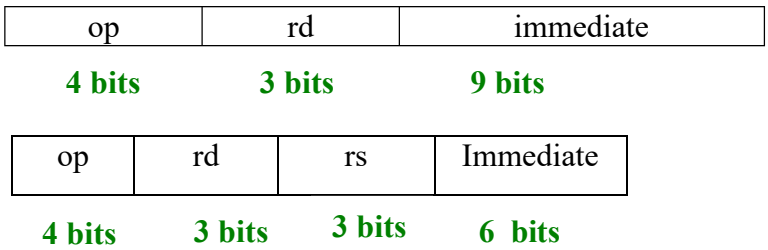


Figure 5: I-format

6.3 J-format

The J-format (jump format) as follows (figure 6):

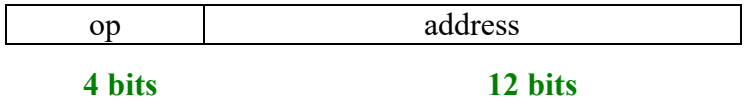


Figure 6: J-format

Table 3 shows the register convention for the BU2020 assembly language.

Table 3: register numbers in BU2020 processor

Name	Register number	usage
Zero	0	The constant value 0
D0 ~ D3	1-4	Data registers
A0 ~ A2	5-7	Address register
SR	8	Status register
BA	9	Base Address
PC	10	Program Counter

7. Hazard solution

You should handle hazard via forwarding

8. The Project Breakdown

The project has three parts:

- In the first part of the project, you should design and evaluate different modules in the data path of the BU2020 processor. You can use VHDL or Verilog language and use a logic simulation tool such as Modelsim to verify your models functionality.
- In the second part you should add the control path to your design and do the simulation and verification process.
- In the third part you should add a memory to your design and write a program test bench. Finally, you should run the program and measure the performance of the processor in term of number of clock cycles and the hardware complexity in term of occupied area.

Best Regards