Computer Architecture Project (CAP)

Project title: Design and implement "CAP23" processor using **pipelined** style
Department: Computer Engineering Department, Iran university of Science and Technology

You should send your VHDL/Verilog codes and also simulation results by the end of July 9 of 2023 and deliver it to TAs

mailto:Hakem.Beitollahi@esat.kuleuven.be

Project description

CAP23 CPU is a 16 bit-RISC processor. CAP23 processor has 5 set of instructions.
- The data transfer-reference set of instructions such as load, store instructions and etc.
- The arithmetic-reference set of instruction such as add, subtract and etc.
- The logical-reference set of instruction such as and, or and etc.
- The conditional branch-reference set of instruction.
- The unconditional jump-reference set of instruction.

1. Register sets

CAP23 has 14 registers. Registers in CAP23 are 16 bits. Each word in CAP23 has 16 bits. Table 1 shows CAP23's registers and their description.

Tabel 1: CAP23 processor registers and their description

Data Registers	D0 ~ D3	4 fast registers to perform arithmetic
Address Registers	A0 ~ A3	4 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	N	C	O	the rest 12 bits

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

Figure 1: Status register

3. Assembly language of CAP23 processor

As it has been mentioned above CAP23 has five assembly categories. Table 2 shows CAP23 assembly language instruction set within their description and also an example for each of them.

Table 2: CAP23 processor assembly language instruction set

Category	Instruction	Example	Meaning	comments
Arithmetic	add	Add d0, d1	d0= d0 + d1	Two operands; overflow detected (opcode =0000)
	add	Add d0, 25	d0 = d0 + memory(BA+50)	Two operands; overflow detected (opcode = 0001)
	subtract	Sub d0, d1	d0 = d0 – d1	Two operands; overflow detected (opcode = 0010)
	Add immediate	Addi d0, 50	d0 = d0 + 50	+ constant; overflow detected (opcode = 0011)
Logical	And	And d0, d1	d0 = d0 & d1	Two reg. operand; logical and (opcode = 0101)
	Shift left logical	Sll d0, 10	d0 = d0<<10	Shift left by constant (opcode = 0110)
Data Transfer CLR MEM OPCODE = 1000	Load word	Lw d0, 7	d0 = memory(BA + 14)	Word from memory to register (28 = 7×2) (opcode = 0111)
	Store word	Sw d0, 10	Memory(BA+ 20) = d0	Word from register to memory (40= 10×2) (opcode = 1001)
	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)

4. Memory access in CAP23

As any word in CAP23 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 addressing is called byte addressing. Figure 3 shows CAP23 memory addresses. For example the address of third word is 8.
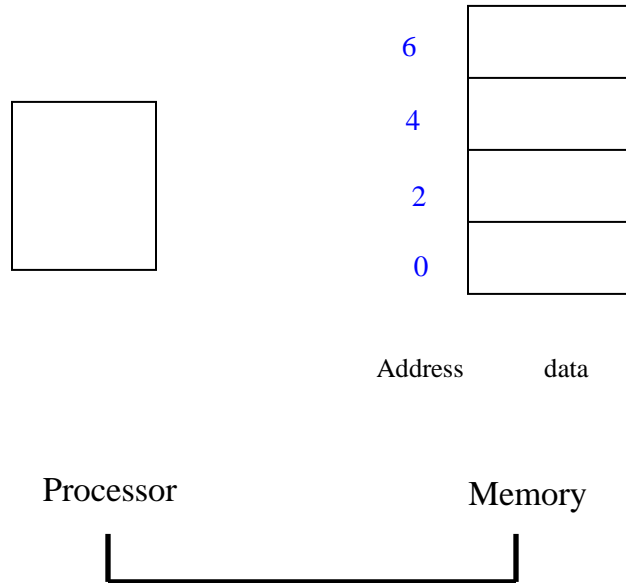

Figure 3: byte addressing in CAP23 processor

5. Types of memory in CAP23

The CAP23 processor has unique memory for both instruction memory and data memory. The CAP23 processor has 4 KB memory. The first 1 KB is used for instruction memory; while the rest of 3 KB memory is used for data memory.
As there is one memory for instruction and data memory, the structural hazard is obvious.

6. Machine code in CAP23 processor

CAP23 has two formats (R-format and I format) for machine code of instructions as follows

6.1. R-format
R-format (Register format) as the follows (figure 4):

op	rd	rs	NOP
4 bits	**4 bits**	**4 bits**	**4 bits**

Figure 4: R-format

❖ op: Basic operation of the instruction, traditionally called *opcode*.
❖ rd: The destination register
❖ rs: The source register
❖ NOP: no operational code (don't care)

6.2 I-format

The I-format as follows (figure 5):

op	rd	immediate
4 bits	**4 bits**	**8 bits**

Figure 5: I-format

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

op	address
4 bits	**12 bits**

Figure 6: J-format

Table 3 shows the register convention for the CAP23 assembly language.

Table 3: register numbers in CAP23 processor

Name	Register number	usage
Zero	0	The constant value 0
D0 ~ D3	1-4	Data registers
A0 ~ A3	5-8	Address register
SR	9	Status register
BA	10	Base Address
PC	11	Program Counter

7. Hazard solution

You are free to handle hazard in any situation you like

8. Input and Output of your program

Input1: a machine code example in a text file (test.txt).

Input2: your CPU has 2 inputs a) clock and b) reset

Output: All important signals (Address of mem, data of mem, PC, status reg, ALU output, etc)

9. Bonus point

Write an assembler, with the help of Python, that translates a sequence of assembly codes to machine codes in a file, then read those machine codes, using your implemented code in VHDL, from the file and execute them.

Success,

Dr. Hakem Beitollahi,