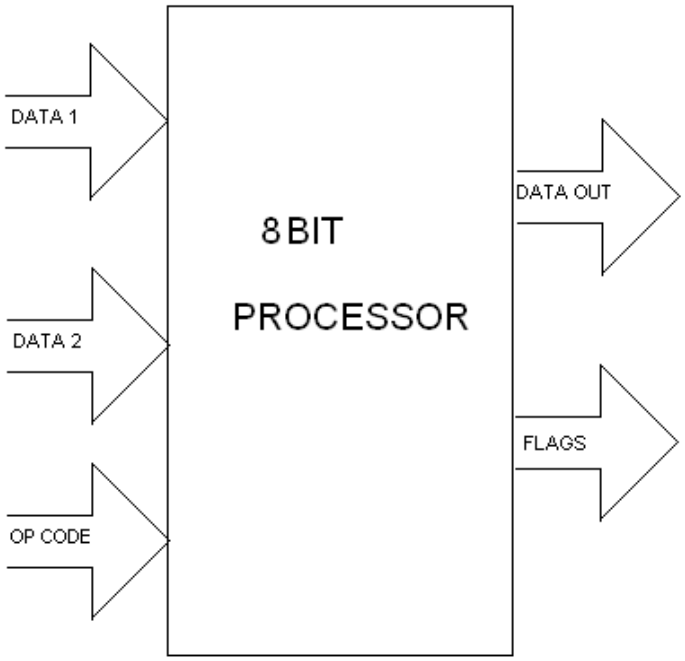
# Hacettepe University
# Department of Computer Engineering
## BBM231 Digital Design Project

**Development Language** : Verilog
**Deadline** : 13.01.2014 - 17:00 is hard deadline.
**Instructor** : Prof. Dr. Mehmet Önder Efe
**Advisor** : R.A. Hüseyin Temuçin
**Submission** : Same as you did in Verilog homework. No paper submissions.

## 1. INTRODUCTION




In this experiment, you will design and implement a simple 8-bit multiprocessor. This processor does not have any support for any memory or I/O component. It has no clock input.

The processor has three input ports: two 8-bit data ports and one 4-bit opcode ports. It has also two output ports: one 8-bit data out port and one 5-bit flags port. The process will have a 5-bit flags register and its value will be given to flags port at the end of processing.

The processor will process the data coming via data ports and will give result to data out port. It also will change the flags via processing. As we mentioned before, the flags register will be given to flags port at the end of processing.

The processor will only process **unsigned integer** values. It has no support for either floating points or negative numbers. The value which comes from ports will have no separated bit for sign. It will hold natural binary number. For example if the value comes from data in port is “00101100”, it will be processed as 44.

The processor will have 5 flags:

Zero Flag, Sign Flag, Carry Flag, Overflow Flag and Compare Flag. They will be detailed later.

## 2. COMMANDS

The processor will support 10 commands:

| OPCODE | COMMAND | RESULT |
|---|---|---|
| 0000 | ADD | DO = D1 + D2 |
| 0001 | SUB | DO = D1 – D2 |
| 0010 | AND | DO = D1 AND D2 |
| 0011 | OR | DO = D1 OR D2 |
| 0100 | XOR | DO = D1 XOR D2 |
| 0101 | NOT | DO = NOT D1 |
| 0110 | CMP | DO = NOT CARE<br>D1 = D2 => ZF = 1<br>D1 < D2 => COF = 0<br>D1 > D2 => COF = 1 |
| 0111 | SHL | DO = D1<<D2 |
| 1000 | SHR | DO = D1>>D2 |

**ADD :** This command is run when the OPCODE is “0000”. Processor adds two data in inports and puts the result to DATA_OUT port at this command. After addition, if there is a carry bit after 8th bit, the carry bit will be set. For example
DATA1 : 10000000 => 120
DATA2 : 10001100 => 140
After processing the out ports will be:
DATA_OUT : 00000100 , CF : 1
00000100 equals to 4 in decimal and set CF means 256. So the result will be
256 + 4 = 260

**SUB :** This command is run when the OPCODE is “0001”. Processor subtracts DATA2 from DATA1and put the result to DATA_OUT port at this command. If the result is negative The Sign Flag will be set. For example :
DATA1 : 00010000 => 16
DATA2 : 10010100 => 20
After processing the out ports will be :
DATA_OUT : 00000100, SF : 1
00000100 equals 4 in decimal and the set SF the result will be -4.

**AND :** This command is run when the OPCODE is “0011”. Processor will bitwise and DATA1 and DATA2 bit by bit and will put the result on DATA_OUT port. For example :
DATA1 : 00010001 => 17
DATA2 : 00010100 => 20
After processing the out ports will be:
DATA_OUT : 00010000 => 16

**OR :** This command is run when the OPCODE is “0100”. Processor will bitwise or DATA1 and DATA2 bit by bit and will put the result on DATA_OUT port. For example :
DATA1 : 00010001 => 17
DATA2 : 00010100 => 20
After processing the out ports will be :
DATA_OUT : 00010101 =>21

**XOR :** This command is run when the OPCODE is “0101”. Processor will bitwise xor DATA1 and DATA2 bit by bit and will put the result on DATA_OUT port. For example :
DATA1 : 00010001 => 17
DATA2 : 00010100 => 20
After processing the out ports will be :
DATA_OUT : 00000101 => 5

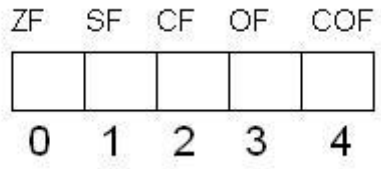
**NOT :** This command is run when the OPCODE is “0110”. This command takes only DATA1 as operand. Processor will bitwise not DATA1 bit by bit and will put the result on DATA_OUT port. For example :
DATA1 : 00010001 => 17
After processing the out ports will be :
DATA_OUT : 11101110 => 238

**CMP :** This command is run when the OPCODE is “0111”. Processor will compare DATA1 and DATA2 and change flag values via compare result at this command.
D1 = D2 => ZF = 1
D1 < D2 => COF = 0
D1 > D2 => COF = 1
Examples
DATA1 : 00010001 => 17
DATA2 : 00010001 => 17
After processing the out ports will be :
DATA_OUT = NOT CARE ZF = 1
---
DATA1 : 00010001 => 17
DATA2 : 00010010 => 20
After processing the out ports will be :
DATA_OUT = NOT CARE COF = 0
---
DATA1 : 00101100 => 44
DATA2 : 00010001 => 17
After processing the out ports will be :
DATA_OUT = NOT CARE COF = 1

**SHL :** This command is run when the OPCODE is “1000”. Processor will shift left DATA1 value DATA2 times and will put the result on DATA_OUT port. The emptied positions will be padded with zero. For example:
DATA1 : 00010001 => 17
DATA2 : 00000011 => 3
After processing the out ports will be :
DATA_OUT : 10001000 => 136

**SHR :** This command is run when the OPCODE is "1001". Processor will shift right DATA1 value DATA2 times and will put the result on DATA_OUT port. The emptied positions will be padded with zero. For example:
DATA1 : 00010001 => 17
DATA2 : 00000011 => 3
After processing the out ports will be :
DATA_OUT : 00000010 => 2

## 3. FLAGS




The processor will have 5-bit flags register and will have five flag values, which are:

- **ZERO FLAG (ZF)** : This flag will be set if the processing result is zero or CMP command value is equal.
- **SIGN FLAG (SF)** : This flag will be set if the result is negative
- **CARRY FLAG (CF)** : This flag will hold carry value for both adding and multiplying
- **OVERFLOW FLAG (OF) :** This flag will be set if the processing result overflow the data out port and carry flag.
- **COMPARE FLAG (COF) :** This flag will be adjust via CMP command :
    - If DATA1 < DATA2, its value will be '0',
    - If DATA1 > DATA2, its value will be '1'
    - If DATA1 = DATA2 its value will be not care '-'

## 4. DESIGN RULES

**When calculating out port values, value conversions from std_logic, bit to other built-in values such as integer or natural is not accepted. You must calculate results with using logic gates. Also usage of arithmetic operators +,-,*,/ is not permitted on signals and ports.**