BIRZEIT UNIVERSITY

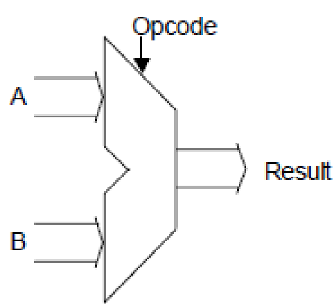
**Faculty of Engineering & Technology**
**Department of Electrical and Computer Engineering**
**First Semester, 2023/2024**

***ENC3310– Advanced Digital Design***
***Course Project***

***Instructor: Abdellatif Abu-Issa & Elias Khalil***

**Overview**
In this project you will build a simple part of a microprocessor. Firstly, you will build two main blocks: the ALU and the register file, then you will connect them together and run a simple machine code program on them.
In order to ensure that your design is related to yourself, various aspects of the design will be dictated by your student ID number. These include the machine code used by your design, the contents of the register file. This means that every student should have a unique design.
You have to work individually in this project and submit a report about your design and implementation of this project.

## Part 1
### The ALU
Write a VERILOG description of an ALU with two 32-bit inputs, *A* and *B*, and a 32-bit output *Result* (module alu).



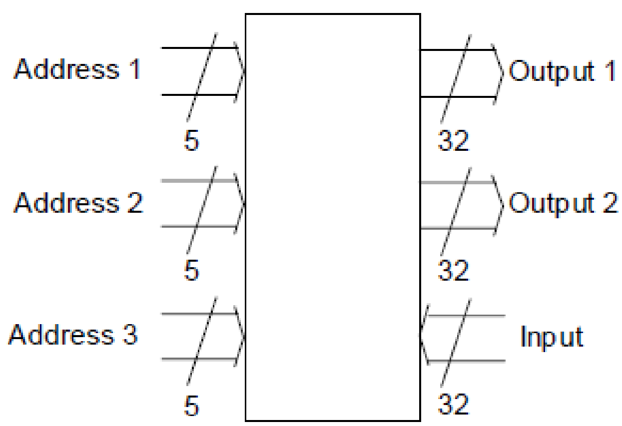

The result is derived from one or both of the inputs according to the value of a 6-bit opcode. The operations that the ALU can perform are listed below:
• a + b
• a - b
• |a| (i.e. the absolute value of a)
• -a
• max (a, b) (i.e. the maximum of a and b)
• min (a, b) (i.e. the minimum of a and b)
• avg(a,b) (i.e. the average of a and b – the integer part only and remainder is ignored)
• not a
• a or b
• a and b
• a xor b

The opcode that will be used to represent each of these operations is determined by the last digit of your students ID number. The table below shows which opcode you should use in your design for each instruction.

| Digit of ID no. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| a + b | 8 | 6 | 6 | 4 | 6 | 3 | 4 | 4 | 1 | 5 |
| a – b | 9 | 8 | 9 | 11 | 2 | 15 | 10 | 14 | 6 | 8 |
| \|a\| | 2 | 10 | 1 | 8 | 5 | 13 | 3 | 8 | 13 | 13 |
| -a | 10 | 12 | 5 | 6 | 4 | 12 | 12 | 11 | 8 | 7 |
| max(a,b) | 12 | 14 | 7 | 13 | 7 | 7 | 7 | 10 | 7 | 3 |
| min (a,b) | 1 | 11 | 8 | 14 | 10 | 1 | 2 | 1 | 4 | 6 |
| avg(a,b) | 13 | 13 | 11 | 7 | 9 | 9 | 6 | 13 | 11 | 10 |
| not a | 5 | 15 | 14 | 9 | 13 | 10 | 13 | 6 | 15 | 2 |
| a or b | 4 | 2 | 13 | 12 | 8 | 14 | 14 | 9 | 3 | 15 |
| a and b | 11 | 3 | 12 | 10 | 1 | 11 | 11 | 5 | 5 | 4 |
| a xor b | 15 | 9 | 4 | 5 | 3 | 5 | 8 | 7 | 2 | 12 |

So, for example if your ID is 122143**8** then the last digit is **8**.
Then a+b is to be represented by opcode 1, a-b is to be represented by opcode 6, |a| is to be represented by opcode 13 and so on. (These are shown as denary (base 10) values; your design may use binary or hexadecimal values.)

**The register file**
Inside a modern processor there is a very small amount of memory that is used to hold the operands that it is presently working on. This is called the *register file*, and normally has the following appearance (module reg_file).




This is a very small fast RAM, typically holding 32 x 32-bit words, and therefore requiring a 5-bit address to select out one of the 32-bit words. It is unlike normal RAM in that it can process three addresses at the same time, two of which are always read operations, and one of which is always written to.

*Output 1* produces the item within the register file that is address by *Address 1*. Similarly *Output 2* produces the item within the register file that is address by *Address 2*. Input is used to supply a value that is written into the location addressed by *Address 3*.

The initial values stored in the register file are determined by the second-from-last digit of your student ID, and are shown in the table below:

| ID/ Location | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 7942 | 11662 | 12642 | 12996 | 4198 | 11930 | 4616 | 15034 | 5986 | 16302 |
| 2 | 13224 | 11562 | 10592 | 11490 | 5596 | 5348 | 11640 | 8854 | 12250 | 2994 |
| 3 | 15462 | 15330 | 6230 | 7070 | 14426 | 7308 | 11254 | 170 | 482 | 1658 |
| 4 | 8026 | 9594 | 8940 | 6026 | 7612 | 15684 | 6786 | 7226 | 14246 | 5474 |
| 5 | 3692 | 14746 | 8776 | 3322 | 6638 | 12346 | 6784 | 4480 | 5124 | 6784 |
| 6 | 9882 | 3288 | 9436 | 10344 | 10040 | 9716 | 12432 | 8928 | 1848 | 10836 |
| 7 | 8248 | 5932 | 3056 | 6734 | 3930 | 7820 | 13548 | 7302 | 5260 | 4648 |
| 8 | 3432 | 1978 | 4850 | 15834 | 4150 | 5190 | 13462 | 8922 | 16170 | 524 |
| 9 | 178 | 4912 | 3406 | 15314 | 6406 | 14702 | 13454 | 1044 | 4766 | 12200 |
| 10 | 2378 | 2380 | 12380 | 6000 | 5400 | 5630 | 11780 | 6706 | 4298 | 3286 |
| 11 | 8302 | 1926 | 548 | 12196 | 8572 | 2352 | 13170 | 258 | 610 | 14734 |
| 12 | 592 | 12726 | 13054 | 11290 | 16324 | 15424 | 2982 | 7354 | 1510 | 10998 |
| 13 | 7430 | 176 | 2800 | 13350 | 8840 | 2670 | 8096 | 3294 | 9794 | 4420 |
| 14 | 10572 | 8408 | 12988 | 2086 | 8258 | 4172 | 514 | 14740 | 7456 | 8754 |
| 15 | 7676 | 8394 | 956 | 6734 | 11228 | 4300 | 3600 | 6532 | 5580 | 3246 |
| 16 | 1238 | 13604 | 2194 | 7430 | 8462 | 4744 | 10870 | 10436 | 9300 | 9040 |
| 17 | 16008 | 10222 | 11914 | 14102 | 13284 | 1286 | 12528 | 11900 | 12314 | 8714 |
| 18 | 2426 | 7262 | 15864 | 13200 | 4676 | 8122 | 9860 | 14694 | 12806 | 12008 |
| 19 | 11930 | 10190 | 11832 | 3264 | 3980 | 4558 | 6166 | 8830 | 10478 | 1006 |
| 20 | 6724 | 8734 | 12346 | 2368 | 5634 | 8534 | 4520 | 8712 | 11556 | 6724 |
| 21 | 12790 | 12432 | 2192 | 15846 | 7632 | 13340 | 14436 | 4532 | 6778 | 12746 |
| 22 | 4842 | 8724 | 1840 | 11710 | 9846 | 6918 | 12136 | 9084 | 8430 | 5462 |
| 23 | 7108 | 5412 | 13996 | 14736 | 5442 | 11700 | 5134 | 13838 | 5700 | 11810 |
| 24 | 6296 | 11082 | 12054 | 5338 | 12488 | 10722 | 11958 | 10018 | 13422 | 7590 |
| 25 | 3322 | 2212 | 4434 | 5544 | 6656 | 3346 | 7688 | 1280 | 11224 | 4368 |
| 26 | 10848 | 6188 | 12152 | 1852 | 832 | 3300 | 5258 | 5814 | 1990 | 10358 |
| 27 | 14698 | 7056 | 9876 | 3898 | 4664 | 2386 | 12420 | 670 | 922 | 15252 |
| 28 | 16378 | 3744 | 8734 | 16252 | 6798 | 11212 | 3560 | 8832 | 6020 | 11954 |
| 29 | 15456 | 5766 | 8308 | 1048 | 14166 | 3504 | 1248 | 15186 | 15768 | 13704 |
| 30 | 1260 | 3412 | 3422 | 5642 | 3246 | 8712 | 8724 | 4512 | 5624 | 1478 |
| 31 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

So, for example, if your students ID is 1225**6**7 then the second-from-last digit of your ID number is **6**. so item 0 should be 0, item 1 should be 4616, item 2 should be 11640, and so on. (N.B. these values are in *denary (i.e. base 10)*. You may need to convert them to binary or hexadecimal.)

**Part 2**

In this part you will connect the ALU with the RAM to form a simple microprocessor.

**Clocking the register file**

The register file that you created in the first part was a combinational circuit. This causes some serious problems if, for example, address 2 and address 3 are the same. The circuit would read output 2 from the location referenced by address 2, at the same time that the input is over-writing that location.

These problems can be solved by synchronising the register file to a clock. You will need to add an extra input named *clock*, and give the register file the following behaviour:

*On the rising edge of the clock:*

- *Output 1* produces the item within the register file that is address by *Address 1*.
- *Output 2* produces the item within the register file that is address by *Address 2*.
- *Input* is used to supply a value that is written into the location addressed by *Address 3*.

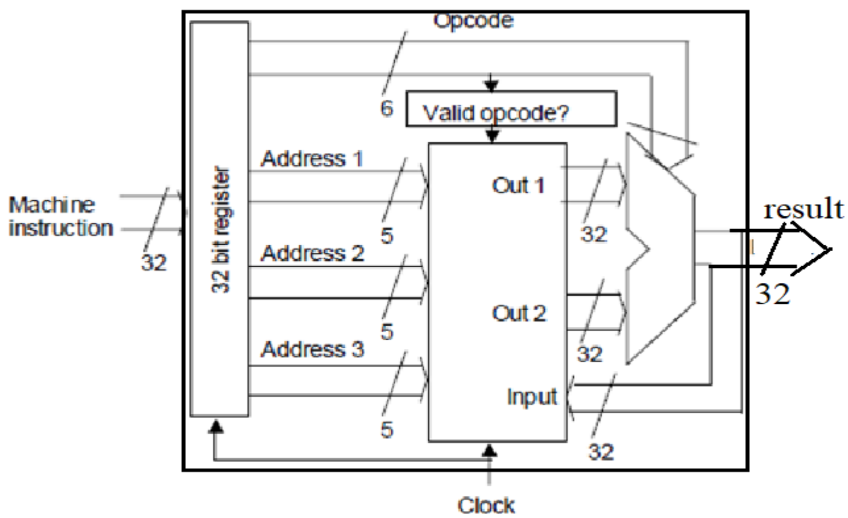
*Under all other circumstances*:
• the outputs are held constant at the values they assumed during the last rising edge of the clock.

You should check your design thoroughly, and in particular make sure that it behaves sensibly when the address for the input is the same as the address of one of the outputs.

**Enabling the register file**
The register file that you created is always sensitive to its inputs, even when the inputs have garbage values. This can cause problems because when the simulation initializes (which corresponds to the real hardware being switched on) all the values of the logic signals initializes to some random garbage value.

Give the register file an enable input. When the enable input=1 the register file will operate normally. Otherwise the register file will ignore its inputs, and will not update its outputs.

**Creating the core of the microprocessor**
Now create a testbench that contains an instance of the register file and an instance of the ALU connected like this (module mp_top):




Machine instructions are supplied to this arrangement in the form of 32-bit numbers. The format of these instructions is as follows:
• The first 6 bits identify the opcode
• The next 5 bits identify first source register
• The next 5 bits identify second source register
• The next 5 bits identify destination register
• The final 11 bits are unused
So, for example, if you want to add the contents of register 1 and register 2 and put the result into register 3, then the machine instruction would be as follows:
• The first 6 bits supply the opcode for the *add* instruction
• The next 5 bits would address register 1, and the next 5 would address register 2
• The next 5 bits address register 3.
• The remaining bits are unused, and should be set to zero.
The *enable* signal to the register should go high when the *opcode* contains a valid value, and should be low otherwise.

Test out your design by supplying machine instructions to it, and check that the operation performed is correct. Make sure you understand the timing of instructions, and in particular the relationship between the clock cycle on which the instruction occurs, and the clock cycle on which the appropriate result is written to register.

**Main Modules templates**

In order to ease the ability to test your design, you need to stick the following modules definition. If you don't stick to the modules definition, then testing your design by the instructor testbench will not be possible and hence you will loose part of the evaluation

```
module alu (opcode, a, b, result );
  input    [5:0]       opcode;
  input    [31:0]      a, b;
  output   reg [31:0]  result;

endmodule
```

```
module reg_file (clk, valid_opcode, addr1, addr2, addr3, in , out1, out2);
  input                clk;
  input                valid_opcode;
  input    [4:0]       addr1, addr2, addr3;
  input    [31:0]      in;
  output   reg [31:0]  out1, out2;

endmodule
```

```
module mp_top (clk, instruction , result );
  input                clk;
  input    [31:0]      instruction;
  output   reg [31:0]  result;

endmodule
```

**Running a program (Testing your design)**

Now create a stream of machine instructions that will act as a small program. The program must act as follows:

1- Create an array of instructions.
2- Include at least one instruction for each of the provided opcodes.
3- Print the input instruction and result (output)

4- For each instruction calculate the expected result, then compare the expected value with your output result.
5- If any instruction fails, then test fails, else test passes

***Synchronising the data***

All parts of the design that are synchronized to the clock should use *only the rising edge* of the clock. If you use both the rising edge and the falling edge, then it's much easier to do the design in simulation, but the real life hardware would be very expensive and complicated to manufacture. You will therefore lose a substantial number of marks if you use both edges of the clock.

**No ops or buffering of opcode**

You may find it useful to create a no-op instruction, i.e. an instruction that instructs the microprocessor to "do nothing" for the next clock cycle, and allocate one of your un-used opcodes to this instruction.
Also, in the first clock, the opcode may arrive before the register values, thus you may need to add buffer (reg) (with one clock cycle delay) for the opcode.

***If you don't manage to finish***

Don't panic. Obviously you will lose marks for having an incomplete design, but you can still achieve a reasonable mark provided that you do a good write up of the parts that you have finished.

## *Writing up the assignment*

### Format of write up

The assignment should be written up as a brief report, which should explain (in about 6-8 pages)

- The ideas behind your design and how it works (in particular what happens on what clock cycle and why);
- How you tested it and how you interpreted the results of your tests;
- Which parts work correctly.

Your Report should be submitted as pdf file while Verilog code should be submitted as .v file (one file for report and one file for the code).
Print outs (screen shots) of your simulation results should be included as appendix 1 (not more than 5 pages; if you have a large number of results, then you should choose only the most meaningful results, and explain in the text what their significance is)

### Deadline and hand-in procedure

The deadline for submission is Wednesday 17/01/2024 till midnight (more than 4 weeks from now). You will lose 10% of your mark for each late day (Late submission is allowed until Thursday 25/01/2024).

### Mark scheme

| | |
|---|---|
| Style, structure and presentation of report | 10% |
| Description of design and testing process | 20% |
| Technical achievements in design, implementation and evaluation | 50% |
| Quality of code (good comments, clear layout, good coding style, meaningful signal and entity names) | 10% |
| Judgment and creativity | 10% |

BIRZEIT UNIVERSITY

# Electrical and Computer Engineering Department

**Project Feedback**
**Course Project (ENCS 3310)**

Instructors : Abdellatif Abu-Issa & Elias Khalil

**Student Name:**…………………………………….. **Student ID:**………………….

Marks

**Report Presentation (10%)**
Language (Spelling and Grammar), style of the report, caption of figures, page numbering…etc.

**Design Process and Outcome (80%)**

- Description of the design and test process **(20%)**
- Technical Achievement in System Design and implementation **(50%)**
- Quality of code **(10%)**

**Judgment and Creativity (`10%)**

Demonstration of good judgment, imagination and creativity in selecting and applying design methods. Good discussion and analysing of the system and suggested improvements.

**Total Mark (Out of 100)**

**Deducted Marks:** late days * 10% per day

***FINAL ALLOCATED MARK (Out of 100)***

***Any evidence for any type of cheating***: ☐yes ☐no