Design, simulate, build, and demonstrate the correct operation of a simple 8-bit RISC processor core along with an external bus to facilitate testing and debugging. This RISC processor has a three-address architecture. The list of commands to be implemented is given below. No interrupts will be available. The instructions must consist of either two, or four 8-bit words. You should develop the instruction formats for your machine (you may borrow ideas from existing processor architectures, like the ARM9). The number of memory cycles to fetch and execute an instruction must be limited to six (six cycles are only needed for load, or store double word from/to memory). The Control Unit FSM should have the following states: Instruction Fetch (IF), Instruction Decode (ID), Operand Fetch (OF), Execute (EX), Write Result (WR). A virtual program/data memory should be implemented on your solderless breadboard using DIP-switches and Bar-LEDs. A single-step logic module should be added to your processor in order to facilitate debugging and the demonstration of the performance of your design.

A tentative functional block diagram is shown on Page 2. You will decide the number of temporary registers needed for your design. Your processor is limited to a single-level on-chip stack. The Branch to Subroutine (BRS) instruction should store the contents of the PC register to the SPC register, the contents of the PSW register to the SPSW register, respectively, and switch from the use of the Set0 Register Bank to the Set1 Register Bank. The Return from Subroutine (RTS) instruction should execute a reverse program context switching. In the PSW register only the following flags are defined: OVF, S, Z, CY and M (Mode). M is 0 when the Set0 Register Bank is active, and it is set to 1, otherwise. You should define the way the OVF, S, Z and CY flags, respectively, are affected by the various commands. All data is assumed in 2's complement representation. In a 16-bit data, the byte at the lower address holds the more significant bits.

List of Commands

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Instruction</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move r1, r2</td>
<td>r2 := r1</td>
<td></td>
</tr>
<tr>
<td>Complement r</td>
<td>r := NOT(r)</td>
<td></td>
</tr>
<tr>
<td>Clear r</td>
<td>r := 0</td>
<td></td>
</tr>
<tr>
<td>Increment r</td>
<td>r := (r) PLUS 1</td>
<td></td>
</tr>
<tr>
<td>AND r3, r1, r2</td>
<td>r3 := r1 AND r2</td>
<td></td>
</tr>
<tr>
<td>OR r3, r1, r2</td>
<td>r3 := r1 OR r2</td>
<td></td>
</tr>
<tr>
<td>XOR r3, r1, r2</td>
<td>r3 := r1 XOR r2</td>
<td></td>
</tr>
</tbody>
</table>
ADD w/o CY \( r_3, r_1, r_2 \)  \( r_3 := r_1 \text{PLUS} r_2 \)

SUB w/o BW \( r_3, r_1, r_2 \)  \( r_3 := r_1 \text{MINUS} r_2 \)

ADD w CY \( r_3, r_1, r_2 \)  \( r_3 := r_1 \text{PLUS} r_2 \text{PLUS CY} \)

SUB w BW \( r_3, r_1, r_2 \)  \( r_3 := r_1 \text{MINUS} r_2 \text{MINUS BW} \)

Logic Shift Left \( r, n \)  \( r := \text{LSL}(r), n \)

Logic Shift Right \( r, n \)  \( r := \text{LSR}(r), n \)

Arithmetic Shift Left \( r, n \)  \( r := \text{ASL}(r), n \)

Arithmetic Shift Right \( r, n \)  \( r := \text{ASR}(r), n \)

Signed Multiply \( r_3, r_1, r_2 \)  \( r_3 := r_1 \times r_2 \) (\( r_3 \) is double register)

Load \( r \) (R0-R7), immediate  \( r := \text{data} \)

Load \( r \) (R8-R15), immediate  \( r := \text{data} \)

Load \( r_1, \text{mem}(r_2) \)  \( r_1 := \text{[mem}(r_2)] \)

Load Double \( r_1, \text{mem}(r_2) \)  \( r_1 := \text{[mem}(r_2)], 16\text{-bit data} \)

Store \( r_1, \text{mem}(r_2) \)  \( \text{mem}(r_2) := (r_1) \)

Store Double \( r_1, \text{mem}(r_2) \)  \( \text{mem}(r_2) := (r_1), 16\text{-bit data} \)

Conditional Branch (F), addr  \( \text{PC} := \text{addr if T =1, PC is updated, otherwise} \)

Branch, addr  \( \text{PC} := \text{addr} \)

Branch to Subroutine, addr  \( \text{SPC} := \text{PC}, \text{SPSW} := \text{PSW}, \text{Set1 is activated} \)

Return from Subroutine  \( \text{PC} := \text{SPC}, \text{PSW} := \text{SPSW}, \text{Set0 is activated} \)

Nop  No operation

For shift operations, \( n = 1\text{-}8 \) (\( n = 0 \) is illegal operation). In case of an illegal opcode, the processor should execute a Nop command.

The timing diagrams for the External Bus Interface are given on Page 3. Use a bus clock signal (BCLK) that is of a divide by 2 derivative of your system clock to generate the bus signals with proper timing. The external input signal ACK# must be sampled in state S2 at the falling edge of the BCLK. If ACK# = 1 at that point then wait states will be inserted between states S2 and S3 until ACK# = 0 is received. In order to facilitate manual demonstration of the performance of the RISC processor a single-step circuit should be added to the design. The single-step waveforms are also given on Page 3.

Map your processor to the Xilinx XC3S500E-FG320-5 FPGA chip (the one that is mounted on the Nexys 2 Board), as well as off-the-shelf ICs and other parts (as needed). Input signals are to be driven by micro switches, output signals are to be displayed by Bar-LED Units on your solderless breadboard. You shouldn’t assume that the input signals to the system are properly synchronized with respect to the Nexys 2 Board system clock signal. You must use the 50MHz SYSTEM_CLOCK signal available on the Nexys 2 Board. You are to provide for your own solderless Bread Board and electronic parts, respectively.
Tasks:

1. Draw a **detailed block diagram and a schematic diagram**, respectively, for the whole system. On the schematic diagram clearly show the partitioning of the system into standard ICs, other parts, and the FPGA chip. (10 pts.)

2. Give the **state transition diagram** for the Control Unit of your processor. (8 pts.)

3. Give the **instruction formats** for all of your instructions along with the action(s) involved. (18 pts.)

4. Use the currently available **Xilinx ISE and the Mentor Graphics ModelSim SE tools** to develop the FPGA segment of the system. You should prove the correctness of your design by **post-route simulations**. Turn in the printouts of your design files (.vhd, .ucf and .do files, as well as sections of your Summary Report with the **resource utilization** and the **device pinout diagram**, respectively), and your **simulation timing diagrams**. **Comment** on your simulation results. You should also include your **conclusions** for the project. (34 pts.)

5. **Build the whole system** using your Nexys 2 Board and your Breadboard and demonstrate its correct operation to the course instructor. The demonstration will be based upon the execution of a program. The program will be situated in a virtual program/data memory chip on your breadboard that will be **emulated** by switches and LEDs. **All instructions** specified must be tested at least once. (30 pts.)

Each team should submit a joint **Project Report**. The project is not done until a **complete** report is submitted. The report should have the main sections as follows: **Introduction, Design, Design Evaluation, and Conclusion**. Your solutions to Tasks 1) to 4) should be represented in the report. **Demonstrations** will be given in the Microcomputer Lab, **Room B-214**.

**Note:** you will lose **10 pts. by each day** your project (a demonstrated circuit along with the report) is tardy. **No** credit will be given if the project is late by **four business days**.