METU EE 446
Computer Architecture
Laboratory

Control Unit Design for
Multi-Cycle CPU

Laboratory Work 5 - Control Unit Design for Multi-Cycle CPU

Objectives

The purpose of this laboratory work is to create a multi-cycle CPU where the previously designed datapath is expanded by the inclusion of a control unit.

The draft instruction set designed earlier will finalized whit the machine code format; i.e the bibnary representations in the machine language and they will be stored in the instruction/data memory to serve for the full execution of a micro-controlled programme. The design of the control unit will be based on the previously implemented datapath architecture that is capable of running exemplar codes written with this ISA.

1 Preliminary Work

To fulfill the requirements of this laboratory work, the following tasks should be performed.

1.1 Reading Assignment

The laboratory manual where the regulations and some other useful information exist is available at the ODTUClass course page. Read that manual thoroughly. If you feel yourself unfamiliar with **multi-cycle CPU architecture**, please refer to the corresponding lecture notes of EE446 course.

1.2 ISA Configuration: From Mnemonics to Machine Code (10% Credits)

For this part, you will complete the design of the instruction set architecture (ISA) that will operate on the designed CPU. In the previous laboratory work, you made a rough clustering of the operations that function in a similar fashion. However; what makes each instruction distinct is the conditions that they satisfy and with operands they use. Indeed, you are expected to have a field for each distinct criterion in the instruction. Therefore; starting by the instruction type field primarily, you may assign binary codes to each instruction in the entire ISA to make them distinguishable among all the others.

The number of bits in a machine code is limited once the datapath architecture is fixed. Therefore, you are expected to squeeze all the conditional information related to the instruction in a minimum number of bits, depending on the previous design of the datapath and keeping the instructions no longer restrains any unnecessary memory utilization. On the other hand, sparse representation of the condition codes and control signals on the instruction itself contributes to the development of a simpler control until with reduced number of functional components. Presumably, it becomes a trade-off between memory utilization and component consumption that the designer is entangled with. An example field assignment for a different processor with 32-bit-long instructions is provided in Figure 1.

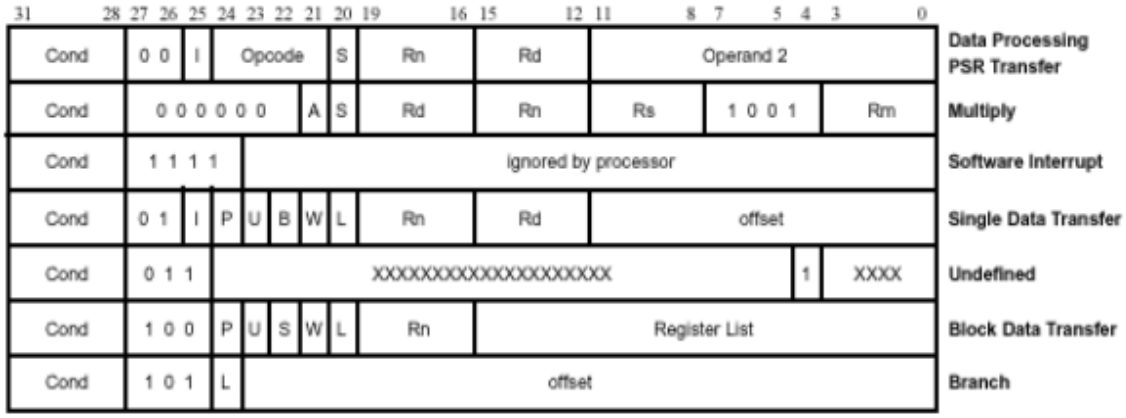

Figure 1: Example Instruction Fields for ARM7 Architecture

In this part of the preliminary work, you are expected to

1. Decide on the field and the associated binary patterns each instruction in the ISA to obtain the machine codes and determine the length of the instructions in your defined ISA.
2. State and defend your motivation in this specific conversion in terms of the above explained trade-off.

1.2.1 Multi-cycle CPU Controller Unit Design (60% Credits)

As explained in the previous manual, a multi-cycle CPU is composed of a datapath, which you have implemented and the control unit you are expected to implement within the scope of this lab. The design of the control unit may be considered as designing the finite-state machine that will generate the control sequence, in the correct conditional and sequential order with respect to the fetch-decode-execute principle.

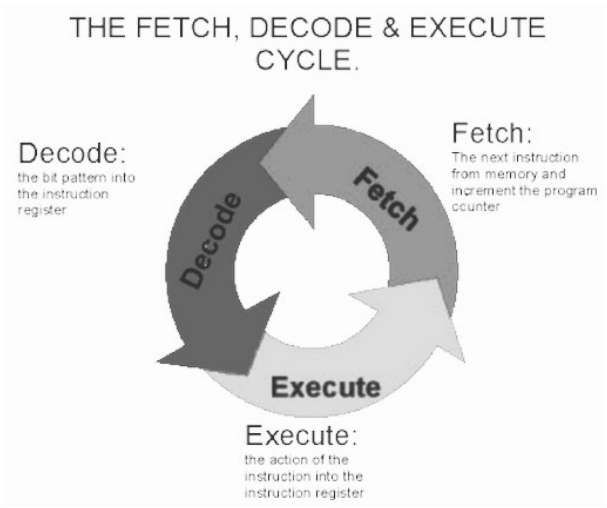

Figure 2: Fetch-Decode-Execute Sequence in Multi-Cycle Operation

To be more broad on the control unit, the above mentioned cycles are to be discussed in detail:

- **Fetch Cycle:** This is the very first cycle corresponding to the operation of a single instruction and it is where the instruction is read from the instruction/data memory to be loaded to an instruction register that holds the current one. Meanwhile, the program counter (PC) is increased to point to the next instruction.
- **Decode Cycle:** Within the decode cycle, the current instruction in the instruction register is decoded to obtain the conditions and the operands.
- **Execute Cycle:** In this final cycle, the desired data manipulation is realized and the corresponding registers and/or memory locations are updated accordingly.

Regarding the above given descriptions, it is required that you determine which control signal in your previous design is to be utilized in which cycle. Intuitively, these cycles are expected to be some states of the above mentioned FSM. For the execution of a complete program code, FSM is expected to function from the first instruction until an END condition is reached. The end condition is most often an END instruction, that you may include in your ISA, that forces the execution to be halted.

For the operation of the FSM, certain external signals, namely **RUN** and **RESET** are also necessary. **RUN**: The external signal that incites the execution of the code from the current instruction (active high).

RESET: Terminates the operation and sets the PC to the very first slot in the instruction/data memory (active high).

In this step of the preliminary work, you are to

1. List the related control signals for the fetch-decode-execute cycles
2. Design an ASM chart for the operation of the FSM as described above. You are not expected to include each instruction separately, but to illustrate the overall functioning of the designed circuit. Assert whether extra states are required.

3. Implement the design of the control unit, by providing a state register to demonstrate the current state and setting up the chunks of circuitry required for each control signal.
4. Write a test bench to simulate the operation of the design and verify its operation through simulations.

1.2.2 Validation of Operation via Microprogrammed Control (30% Credits)

Now that you have completed the implementation of the multi-cycle CPU, it is required to verify its operation through microprogrammed control. These microprograms will be some small test codes that you will write to test the full execution of a code. along with the main codes to call these, you are supposed to

1. Write a subroutine that get an 8-bit number and computes it 2's complement.
2. Write a subroutine that computes the sum of an array of numbers and stores it in a memory location. The length of the array is provided in a memory location, as well as the array itself.
3. Write a subroutine that determines the evenness/oddity of a 8-bit number, $n_7n_6n_5n_4n_3n_2n_1n_0$. If odd, the expected output is $n_4n_3n_2n_10n_7n_6n_5$, otherwise, $0n_4n_3n_2n_1000$.

After the codes are completed the verification of the design will be first done on the test bench, then the design will be embedded to DE0-Nano for device operation. In brief,

1. Write your routines and convert them into machine codes, then store them into the instruction/data memory to be executed.
2. Test the operation of your codes with the test bench module that you have prepared in part 1.2.1.
3. Define a register to be used for visual purposes and store all the results to data register. Connect the LEDs of the FPGA to this register and export the control external signals RUN and RESET to be controlled by the switches for demonstration. CLK can be provided externally using the push button on the DEEs.
4. Test the operation of the entire design to verify its validity.

2 Experimental Work

The operation of the designed multi-cycle CPU is to be verified and demonstrated to the lab instructor in each of these steps.

1. Test the implemented FSM design for the controller unit using through ModelSim. Verify correct operation for a single instruction as well as the capability to execute a whole sequence.
2. Embed your design on DE0-Nano and provide each program, as machines codes, separately, in the instruction/data memory to testing. Verify the proper execution of each of the programs and demonstrate the result to your lab instructor.

!!! Important Notice !!! When using DEES in your experiments, **DO NOT CONNECT VCC pins of DE0-Nano Board** to DEES's VCC terminal. Also, to make a common voltage reference, CONNECT GND of DE0-Nano Board to DEES's GND terminal. Please make sure that you turn all supplies OFF while making any connection throughout laboratory work.

3 Parts List

```
DE0-Nano Board
DEES
Oscilloscope
```