Laboratory 9

9. Pipeline MIPS CPU Design: 16-bits version

9.1. Objectives

Study, design, implement and test

- **MIPS 16 CPU, pipeline version**

Familiarize the students with

- Pipeline CPU design
- Xilinx® ISE WebPack
- Digilent Development Boards **(DDB)**
  - Digilent Basys Board – Reference Manual
  - Digilent Basys 2 Board – Reference Manual
  - Digilent Basys 3 Board – Reference Manual

9.2. Transforming the MIPS 16 Single-Cycle CPU to a Pipeline CPU

! You must attend/read lecture 8 in order to fully understand the Pipeline CPU

Remember that an instruction execution cycle (lecture 4) has the following phases:

- IF – Instruction Fetch
- ID/OF – Instruction Decode / Operand Fetch
- EX – Execute
- MEM – Memory
- WB – Write Back

The data-path of the single-cycle processor (32-bit version), including the control unit and the necessary control signals, is presented in the next figure. In order to reduce the complexity of the data-path the control signals were not explicitly connected, but rather they can be easily identified by their names.

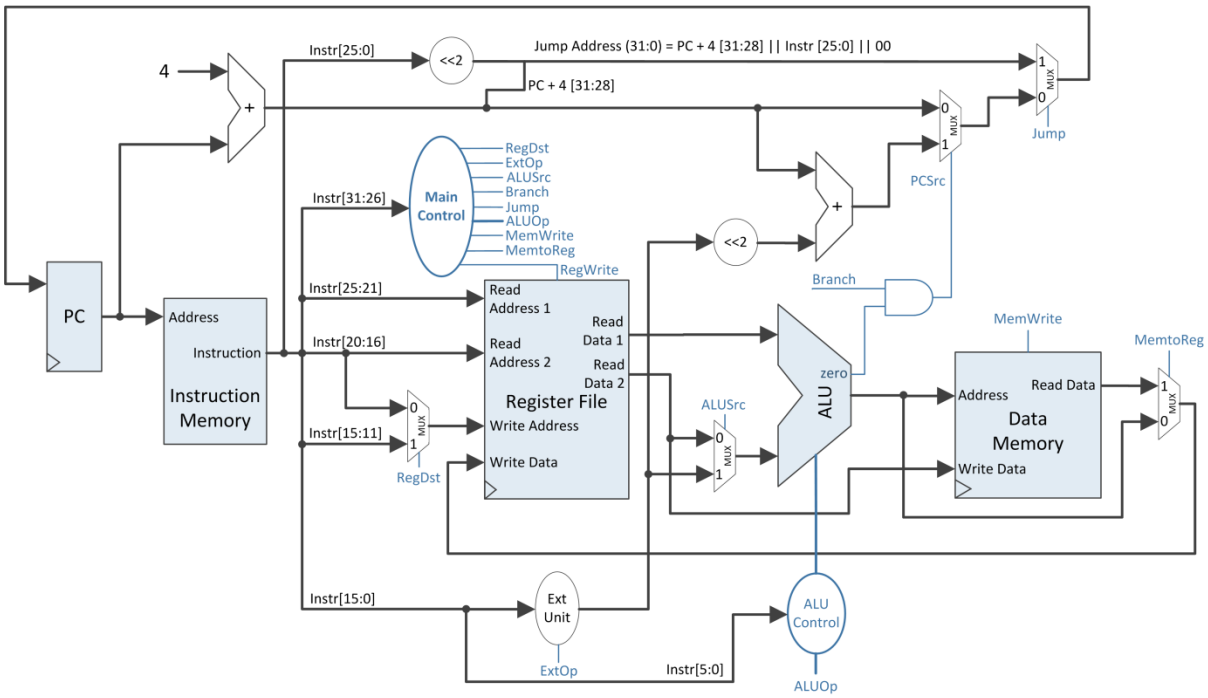

Figure 9-1: MIPS 32 Single-Cycle Data-Path + Control

As a reminder, the instruction formats for your MIPS 16 processor are presented below:

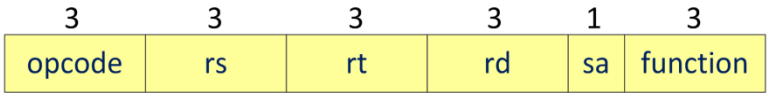

Figure 9-2: R-type Instruction format

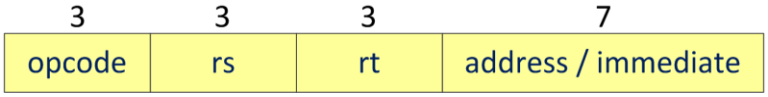

Figure 9-3: I-type Instruction format

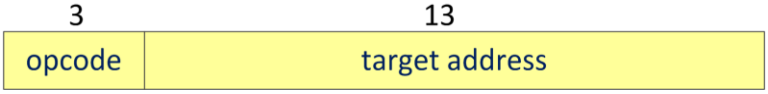

Figure 9-4: J-type Instruction format

The main issue with the single-cycle MIPS CPU is the length of the critical path, for the load word instruction (see lecture 04). The necessary time for transmitting the data along the critical path must be covered by the clock cycle time. This results in a long cycle time (slow clock).

In order to reduce the clock cycle time, the solution is to partition the data-path along the critical path with rising edge triggered registers (D flip-flops). These registers are inserted between the MIPS 32 functional units that coincide with the instruction execution phases: IF, ID, EX, MEM, WB. In this manner, one can simultaneously execute at most 5 instructions, each of them executing one of the five execution phases. The pipeline execution units are also referred to as **stages**.

The data-path together with the control unit for the pipelined MIPS 32 CPU is presented in Figure 9-5.

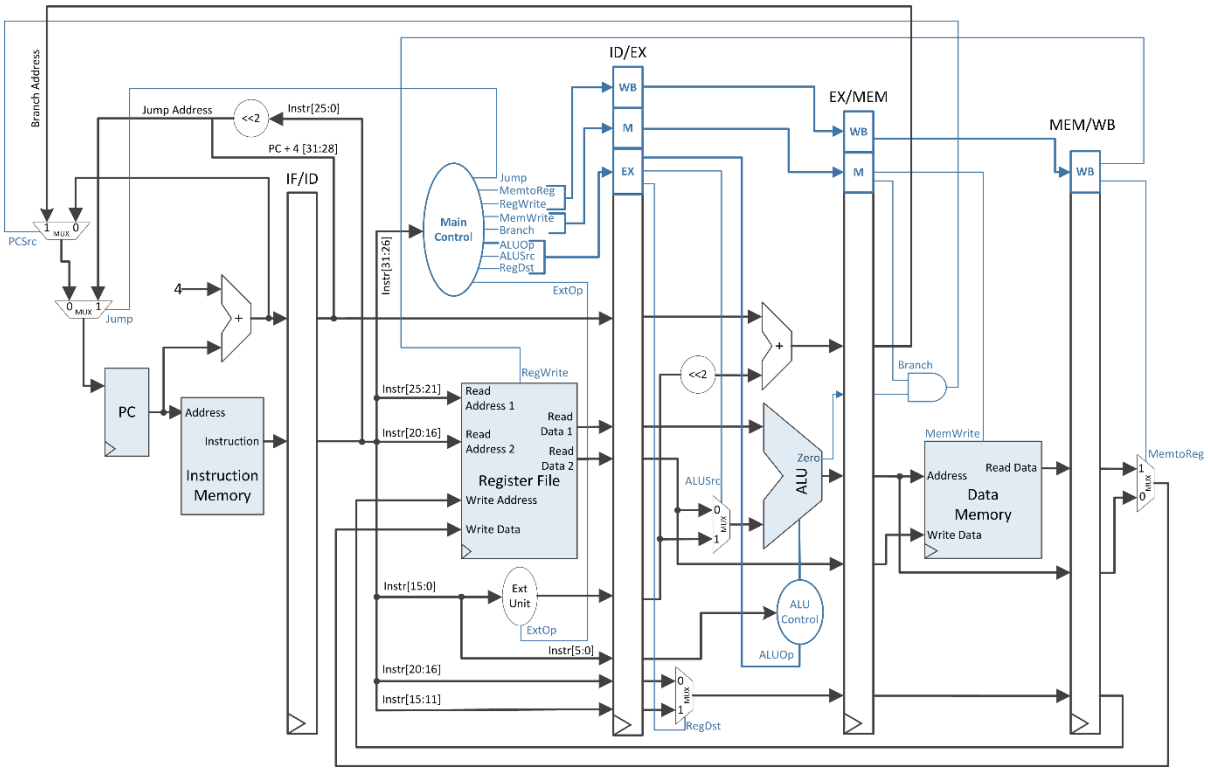

Figure 9-5: MIPS 32 Pipeline Data-Path + Control, obtained from the partitioning of the Single-Cycle Data-Path

Each intermediate register will be referred depending on its position between the pipeline stages. The register between the IF stage and the ID stage is IF/ID, the one between ID and EX is ID/EX, etc.

The role of these intermediate registers is to hold the intermediate results of the instruction execution in order to provide these results to the next stage, in the next clock cycle.

Furthermore, the execution on the data-path depends on the control signals values, which are specific for each instruction. So, through the intermediate registers (starting with the ID/EX register) the control signals will also be provided for the next stages. The control signals are symbolically grouped after the stage name where they belong.

The control signals are transmitted together with the intermediate results until the stages where they are needed.

Lecture 8 explains in more detail the design of the pipeline CPU; the details presented so far represent the necessary knowledge for transforming your own MIPS 16 single-cycle CPU into a pipeline one.

One notable difference between the two data-paths is that the multiplexer used for selecting the write address for the Register File is placed in EX, not in ID as in the single-cycle CPU case. There are 2 possible solutions:

a) Leave it in the ID stage. In this case, the RegDst signal will not be transmitted through ID/EX and will be connected directly from the control unit.
b) Move it to the EX stage, modifying the input / output ports of the ID and EX units, and transmit the RegDst signal according to the presented pipeline data-path.

Observation: The MemRead signal will be ignored, as in the single-cycle case.

9.3. Laboratory Assignments

Read carefully and completely each activity before you begin!

Prerequisites:

- Xilinx project with “test_env” including the complete and correct implementation of the single-cycle MIPS 16 CPU.

9.3.1. Verify the MIPS 16 CPU design

Before you begin transforming your single-cycle processor into a pipeline one, generate the *.bit file and investigate the clock frequency of your processor: **Processes → Design Summary/Reports.** Open **Detailed Reports → Synthesis Report.** In the synthesis report, locate the section about the clock frequency, similarly with the following text:

TIMING REPORT
...
Timing Summary:
…
  Minimum period: 13.284ns (Maximum Frequency: **75.280MHz**)
…

Write down the frequency of your single-cycle processor, so that you can compare it with the frequency of the pipeline version.

If you have not completed the testing of the single-cycle processor, you must complete it now, before you begin the pipeline implementation.

The pipeline implementation must start from a fully functional single-cycle MIPS 16 version.

9.3.2. Design of the intermediate registers (paper and pencil)

For each intermediate register identify the fields that it must store, taking into account your own MIPS 16 implementation.

Use the data-path from Figure 9-5 (!) but keep in mind your MIPS 16 processor's features: 16 bits, not 32; based on your own chosen instructions the data-path may contain additional elements.

For example, for the first intermediate register, IF/ID, one must memorize the following two fields (generic name according to the stage they belong):

- IF.PC+1 – 16 bits
- IF.Instruction – 16 bits

It results that the IF/ID register should contain 32-bits.

Similarly describe the ID/EX, EX/MEM, MEM/WB registers.

When describing the fields of the next intermediate registers, take a look at the associated functional units (the inputs unit and the destination unit respectively: example for IF/ID the IF and ID units respectively) in the laboratories 5, 6, 7 and 8. Identify the fields from the input/output ports of the functional units. This step will be used in the next assignment.

9.3.3. Describe the intermediate registers in VHDL

For this assignment, you will work in the "test_env" entity (where the components of the processor are instantiated and connected together).

Attention: When introducing the new pipeline registers, some small modifications in your functional units may appear. You will easily identify these modifications by carefully studying Figure 9-5 and the particular data-paths for each functional unit of the MIPS 16 processor – laboratories 5, 6, 7, 8. For example, the ID unit needs an additional input port for the write address of the Register File, address that will come from the last pipeline register MEM/WB.

You will not declare new entities for the pipeline registers. For each register, you have to declare a signal of appropriate length (according to the previous assignment) and the behavior of the pipeline register will be described with one process (synchronous data transfer on the rising edge of the clock signal). In order to ease the testing of the processor each register will be controlled with an enable signal (the same MPG output used for validating the writing in the PC register).

You will realize the partitioning of the single-cycle data-path with the pipeline registers and make the necessary correct connections in the mapping of the functional units.

For example, the value of the RegWrite control signal will be transmitted through the MEM/WB register and will be mapped at the input of the ID Unit.

Pipeline register example (one can also use concatenation): for IF/ID you must declare a 32-bit signal RegIF_ID and describe the behavior of the register in one process:

On rising edge of the clock
RegIF_ID(31..16) <= PC+1;
RegIF_ID(15..0) <= Instruction;
Where PC+1 and Instruction are the outputs of the IF unit.

Alternatively, you can declare new signals and concatenate the stage name to the signal name like:
On rising edge of the clock
PC_ID <= PC + 1;
Instruction_ID <= Instruction;

9.3.4. Homework – Paper and Pencil

Draw your own MIPS 16 pipeline CPU data-path together with the control unit and control signals.

9.4. References

[1] Computer Architecture Lectures 3, 4 & 8 slides.
[2] D. A. Patterson, J. L. Hennessy, “Computer Organization and Design: The Hardware/Software Interface”, 5th edition, ed. Morgan–Kaufmann, 2013.
[3] D. A. Patterson and J. L. Hennessy, “Computer Organization and Design: A Quantitative Approach”, 5th edition, ed. Morgan-Kaufmann, 2011.
[4] MIPS® Architecture for Programmers, Volume I-A: Introduction to the MIPS32® Architecture, Document Number: MD00082, Revision 5.01, December 15, 2012
[5] MIPS® Architecture for Programmers Volume II-A: The MIPS32® Instruction Set Manual, Revision 6.02
[6] MIPS32® Architecture for Programmers Volume IV-a: The MIPS16e™ Application-Specific Extension to the MIPS32™ Architecture, Revision 2.62.
- Chapter 3: The MIPS16e™ Application-Specific Extension to the MIPS32® Architecture.