Summary of Chapter 4 (partial)

Marcus van Bergen

15-September-2014

Abstract

This summary will be from pages 244 - 303

4.1 - The introduction This paragraph will show parts of the MIPS instruction set. Some of these being:

- Memory reference instructions SW:(Store word) and LW:(Load word)
- Arithmetic instructions
- The branch instruction (beq) and jump (j)

Throughout this chapter we will be implementing many of the basic design principles that we learned in chapter 1.

For all MIPS instructions two steps are always followed

- Bringing the PC (program counter) to the memory and get the instruction from the memory.
- Read one or two registers, and using the instruction to understand which registers need to be read.

Simplicity is favored with MIPS processor because the third step is mostly the same, even with there being 3 different types of instructions.

Figure 4.1 makes it seem as if data can come from more than one place to for example the PC. But in reality, there must be chosen between which line it comes from. For this a multiplexor is used; in this case it is referred to as a data selector.
A second multiplexor will be introduced, a control unit. Its job is to determine how to set the control lines for the functional units.
A third multiplexor which will handle beq instructions, adding 4 to the PC...

4.2 Logic Design Conventions Data path elements in MIPS can be data values and elements that contain state. Those which are both are called: combinational. Given the same input they always produce the same output. EG: ALU or AND gate.
A state elements are things that have an internal memory. Therefore they are called "State" elements. Restarting the computer would have no effect on their memory. State elements require two inputs: write into the clock and element. A state elements also needs one output, to show what was previously written.
Components that require both an input and output are called sequential.

It is important not to read and write data at the same time, to make sure variables don't get the wrong value. For this we use a clocking mythology, to determine when it is time to read and time to write data.
We will use an edge-triggered clocking scheme, which states all changes will occur when at the clock edge.

When a state element is not written to, there is a control signal needed. If the control signal is asserted (true) then change will happen to the state element. If deasserted, it will not.

4.3 - Building a Datapath To start building a datapath we need to things:

- A datapath element is an element within the MIPS processor designed to operate or hold data.
- A PC (program counter)

R type operations are known as operations that use the ALU. A processor's 32 registers are stored in what is known as a: register file. This register file contains the state of the computer.

Sign extend: is to make the size of the 16 bit offset field larger by replicating the high-order sign bit of the original data.

Branch targeting address: Finding the specific branch address. This is done mostly by adding the PC to the 16 bit offset.

Branch taken: is when the branch condition is true, and the program counter becomes the branch target
Branch not taken: is where the branch condition is not true, and the program counter becomes the address of that instruction.

4.4 - Simple Implementation Scheme This chapter teaches us how to create a Single clock cycle implementation, with a set by set explanation on all the topics previously discussed. In reality this is not used, due to it being too slow.

Truth table: is a logically designed table which shows us all the values of the inputs and what the resulting outputs would be.

Don't care term: Is when the output is not dependent on inputs. Represented in truth tables with an "X".

Opcode: Specifies what piece of the operation will be done.

4.5 - An overview of Pipelining Pipelining is a technique of execution of instructions, so that they are overlapping each-other with hopes to be more efficient than without pipelining. In principle, instead of trying to increase the speed of execution, pipelining tries to increase throughput of instructions. With MIPS processors, we try to pipleline the 5 classic instructions:

1. Fetch instruction from memory
2. Read registers while decoding instruction (standard two in one)
3. Execute the operation or calculate address
4. Access an operand in data memory
5. Write the result into the register

We can use the below formula to represent Time between instructions:

$$Timebetweeninstructions_{Pipelined} = \frac{Timebetweeninstruction_{Non-pipelined}}{Numberofpipestages}$$

Structural hazard: Is when the planned instruction isn't able to be executed due to hardware not being able to support this combination of executions.

Data hazard: Is simply when a planned execution cannot be executed due to the required data not being there yet.

Forwarding/Bypassing: To partially solve Data Hazards, the required data will be requested from internal buffers, rather than wait for the program to give out the data.

Pipeline Stall: AKA a bubble, is a stall which is done to resolve a hazard.

Control Hazard: Is the third type of hazard that can come forth with pipelining. This happens when the order of the information to come in is not what the computer expected. To resolve this, the computer can do two things: 1.Stall 2.Predict

Branch Prediction: Continuing the pipeline at full speed, the computer will assume an outcome to continue pipelining, and continue to change the guesses depending on their success.

4.6 - Pipelined Datapath and Control This section will go deeper on how it is possible to pipeline a Datapath (4.4) using 5 stages. The pipleline technique breaks the 5 stages over the architecture of the entire MIPS processor.

You might ask, how is it possible to then call on data or read in data? The program counter acts as a sort registerfile, in this case called a pipeline register. This allows for updates results to get stored in the PC and further the pipeline process.

In order to get some form of control, the pipeline registers will be expanded to fit control information. The controlling process starts at EX because stages 1 and 2, don't have much controlling needed (no calculations) and from after that every stage gets it's own control information in the pipeline registry.

1. IF: Instruction fetch
2. ID: Instruction decode and register file read
3. EX: Execution or address calculation
4. MEM: Data memory access
5. WB: Write Back

End of the summary.