

**Department of Electrical and Computer Engineering
State University of New York at Stony Brook**

ESE 555 Advanced VLSI Systems Design (Fall 2009)

CAD Assignment 6: Program Counter and Registers

Assignment

To design the program counter (PC), Instruction Register (IR), and Data Address Register (DAR) units for your microprocessor.

Description

PC

All instructions start by using the contents of the program counter as the address to fetch the next instruction. The program counter stores the current instruction address and calculates the address of the next instruction. Storing the current instruction address requires nothing more than a resettable 16-bit register. However, depending on the current instruction, we must also determine whether the next instruction is the instruction that follows sequentially or the instruction resulting from a jump or a branch.

Except on Jumps and Branches, the next instruction follows sequentially from the current instruction (i. e., $PC \leftarrow PC + 1$). The following table lists the instructions from the baseline architecture, which may force an address different from the next sequential address in the PC.

INSTRUCTION	NEXT PC VALUE
Bcond disp	$PC \leftarrow PC + \text{disp}$ (sign ext.) or $PC \leftarrow PC + 1$
Jcond Rdest	$PC \leftarrow R_{\text{dest}}$ or $PC \leftarrow PC + 1$
JAL Rlink, Rdest	$R_{\text{link}} \leftarrow PC + 1$ $PC \leftarrow R_{\text{dest}}$

A complication is presented by the fact that the processor is pipelined. One must decide how to handle changes in program flow in general, as the change in program flow happens, in our case, in the second stage. Meanwhile, unless we design the control to do differently, the processor would continue to fetch and decode subsequent instructions that follow the branch or jump. Several possible solutions are:

1. Always follow a branch or jump instruction with a NOP, or with other instructions that should be executed anyway. This is done by the compiler in many RISC processors. This is the simplest solution from a hardware point of view. Often the branch delay slots can be filled with useful instructions.
2. Lock the first stages of the pipeline so that they do not continue to fetch and decode instructions following a jump or branch. This is equivalent to forcing NOPs in the hardware. This approach adds some complexity, while reducing code size somewhat, and reducing throughput somewhat.
3. For conditional branches, guess whether program flow will be changed or not, and continue fetching and decoding instructions. This requires a fast decode (to know whether the instruction is a conditional branch or jump), a separate arithmetic unit to calculate PC values, and the ability to squash instructions in the pipe if the guess was wrong (this is found out when the instruction gets to the second stage).

Option 1 is acceptable for the baseline machine.

You should be able to initialize the register to a known state, though this state may not necessarily be 0x0000, depending on your application. For example, sometimes processors have operating modes such

that they come out of reset fetching from internal memory in one mode and from external memory in another mode.

Implementation

Register Implementation

From Table 1 it is clear that the next PC value to be stored could be from either an incrementer (normal), the register file (Jump), or an adder (Branch). The PC unit block shown in your baseline architecture block diagram, is assumed to include an adder. Branch and Jump instructions could alternatively use the ALU to calculate the next address. It would also be possible to use the ALU to increment the PC in the normal case, too, thereby reducing chip area, but complicating control and routing, and lengthening the critical path. The method implied by the bus interconnections in the baseline architecture block diagram is shown below in Figure 1.

Requirements

- A file describing the approaches you use to implement the PC and why you used those approaches.
- Schematics for the Program Counter with all necessary combinational logic to implement all the types of next address calculation.
- Layout of the PC.
- Spectre traces showing delays. Describe how you arrived at the delays, referencing any schematics you created solely for the purpose of delay calculation.
- DRC, LVS reports for the PC.

Deadline

November 24