

GEORGIA INSTITUTE OF TECHNOLOGY

College of Computing

CS6290/CS4290 — High-Performance Computer Architecture

Fall 2000

CS6290/CS4290
Homework 6

Issued: October 29, 2000
Due: November 3, 2000



Revised 30-Oct-00

Purpose: This homework covers prediction and software support for speculation.

Reading: H&P Sections 4.3, 4.6
[Emer97], Itanium paper

Problems:

1. Mispredict Penalty.
2. Predictor Circuits.
3. Reading: Merced/Itanium paper.

Collaboration: (*As in the syllabus*) collaboration on projects and homework in **pairs** is encouraged. If you work in a pair, turn in one write-up with the names of both collaborators. You're welcome to discuss high-level concepts with other groups, but all homework solutions must be worked out and written up separately.

Problem 1: Mispredict Penalty

A: Problem 4.10 in the book.

Problem 2: Prediction Circuit

Here's a predictor described using the notation from [Emer97]. I've extended the notation slightly with a `define` keyword meant to distinguish between definitions and instances: a definition just describes how to build a structure. An instance is the real structure.

The variables `PC`, `DIR` and `TAKEN` come from the processor state: the program counter, the direction of the branch instruction (forwards or backwards) and whether the branch was ultimately taken or not.

```
/*
 * first, a familiar definition: an N-bit predictor array.
 * The 'P' inside the definition refers to the results of the
 * prediction. The ++ and -- operators above are defined to be
 * saturating arithmetic with nbits of precision.
 */
define nbit[nbits, depth](index, feedback)
    = MSB(P[nbits, depth](index, (MSB(P) == T) ? P++ : P--));
    /* ^ */
□          /* 30-Oct-00: c-like syntax ^ */
/*
 * second, three 'mystery' definitions of structures.
 */
define foo[](index, feedback)
    = P[1, 256](index, feedback);
define bar[](VOID, feedback)
    = P[8, 1](0, (P << 1) | feedback);
define baz[](index, feedback)
    = P[8, 256](index, (P << 1) | feedback);
□          /* ^^^^^^^ */
    /* 30-Oct-00: c-like syntax for concatenation of bits */
/*
 * third, the actual instance of the predictor. It includes instances
 * of all the definitions above.
 */
prediction = nbit[3, 256]((PC
    xor bar[](VOID, TAKEN)
    xor baz[]((PC << 1) | foo[](PC, DIR), TAKEN)),
□          TAKEN);          /* ^ */
    /* 30-Oct-00: missing paren ^ */
```

A: Draw and label the circuit described by the notation above. Label all storage arrays with sizes and all wires with widths.

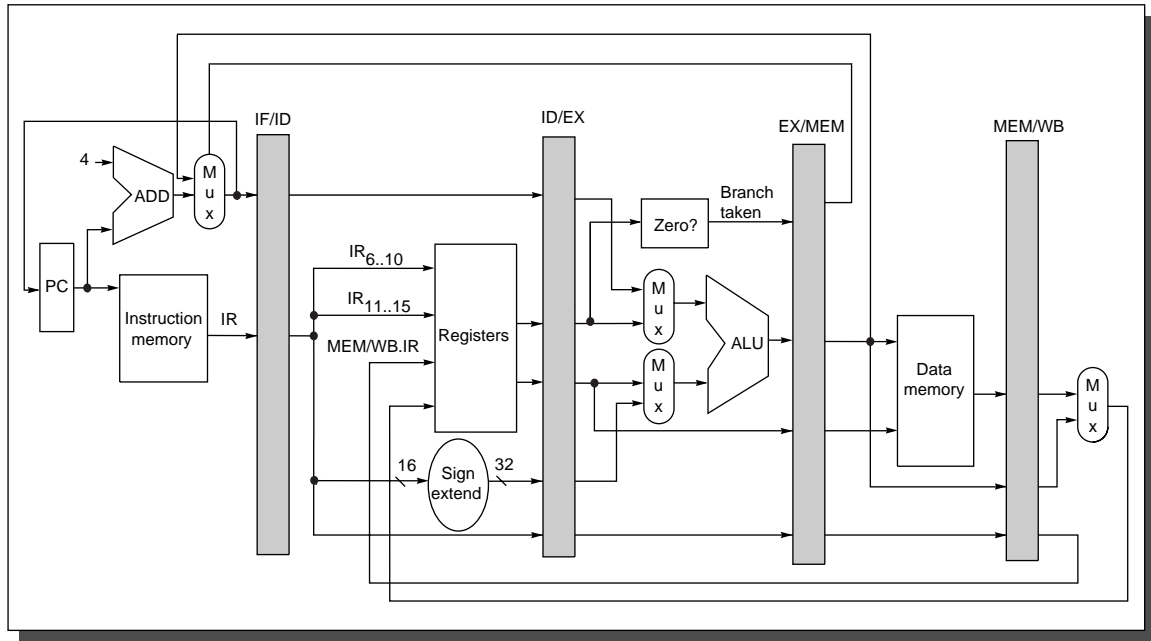


FIGURE 3.4 The datapath is pipelined by adding a set of registers, one between each pair of pipe stages.

B: Referring to Figure 3.4 in the text (reproduced above), where do the PC, DIR and TAKEN signals come from? Indicate with wires drawn on the diagram. Copies of the diagram are available online if useful.

C: What does the `foo[]` structure do in the in predictor? Why is it needed?

D: What information does the `bar[]` structure provide in the predictor?

E: What information does the `baz[]` structure provide in the predictor?

Problem 3: Reading

Read the paper on the Intel Merced/Itanium. The Itanium eschews dynamic (out-of-order) scheduling in favor of static scheduling in software by a compiler. That means that any speculative execution has to be described to the hardware statically by the compiler. What features does the Itanium provide to support speculation by the compiler?