**Question #1**

Let consider the Branch Prediction Mechanism based on the Branch History Table (BHT).

You are requested to

1. Explain the reasons why a Branch Prediction Unit is important in pipelined architectures
2. Describe the general architecture and behavior of a BHT
3. Detail the architecture and behavior of a BHT using a 2-bit saturating counter
4. Detail the architecture and behavior of a BHT using a (2,2) correlating predictor.

The Branch Prediction Unit is important to avoid Control Dependencies.

It can reduce the performance losses by predicting how branches will behave.

Branch History Table is an Hardware Table composed by N rows of M bits, usually 1 or 2 bits.

When a branch is decoded the log2(N) lower bits (except for the bits for instruction alignment) of address of instruction are used to access in BHT . The BHT predicts the branch basing on the previous branch. As soon as the result of branch is known the BHT is updated.

In this case the BHT have for each entry a 2-bit suturing counter. Every time that the result of a branch is taken, the BHT increase the counter, otherwise, if the result of a branch is untaken the counter is decreased. The BHT entry predict taken or untaken basing on 2-bit suturing counter, it predicts taken if the counter is 2 or 3, it predict untaken if the counter is 0 or 1.

**Question #1.2**

Let consider the issue of predicting branch results.

You are requested to

1. Explain why this issue is important in pipelined architectures

2. Summarize the characteristics of Static and Dynamic Branch Prediction mechanisms,

highlighting their main advantages and disadvantages

3. Describe the general architecture and behavior of a BHT

4. Detail the architecture and behavior of a 1-bit BHT.

Predicting branch is important in pipelined architectures to avoid control dependence, because in pipelined architectures when a branch instruction is in decode stage the next instruction is fetching, but the processor doesn’t know if the branch is taken or not. This can lead to a loss of performance.

In Static mechanism the prediction is handled by compiler, there are some different methos:

1. Predict always Taken, if we know in advance the target address
2. Predict always untaken
3. Prediction based on direction, if the target is forward predict untaken, otherwise taken. This is due to the behavior of loops.
4. Predict on the basis of profile information

Dynamic methos are completely handled by a dedicated hardware like, BHT or BTB.

Static solutions don’t need a more complex processor, but they achieve a lower performance.

**Question #2**

Let consider a MIPS64 architecture including the following functional units (for each unit the number of clock periods to complete one instruction is reported):

* Integer ALU: 1 clock period
* Data memory: 1 clock period
* FP arithmetic unit: 2 clock periods (pipelined)
* FP multiplier unit: 4 clock periods (pipelined)
* FP divider unit: 8 clock periods (unpipelined)

You should also assume that

* The branch delay slot corresponds to 1 clock cycle, and the branch delay slot is not enabled
* Data forwarding is enabled
* The EXE phase can be completed out-of-order.

You should consider the following code fragment and, using the table in the following page (where each column corresponds to a clock period), and determine the pipeline behavior in each clock period, as well as the total number of clock periods required to execute the fragment, reporting the result in the right column in the table below. The value of the constant k is written in f5 before the beginning of the code fragment.

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MIPS64 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; for (i = 0; i < 10; i++) {

; v4[i] = v1[i]/v2[i] + v3[i]\*k;

; }

|  |  |  |
| --- | --- | --- |
| .data | comments | Clock cycles |
| V1: .double “10 values” |  |  |
| V2: .double “10 values” |  |  |
| V3: .double “10 values”  V4: .double “10 values” |  |  |
|  |  |
|  |  |
|  |  |
| .text |  |  |
| main: daddui r1,r0,0 | r1← pointer | 5 |
| daddui r2,r0,10 | r2 <= 20 | 1 |
| loop: l.d f1,v1(r1) | f1 <= v1[i] | 1 |
| l.d f2,v2(r1) | f2 <= v2[i] | 1 |
| l.d f3,v3(r1) | f3 <= v3[i] | 1 |
| mul.d f5,f3,f4 | f5 <= v3[i]\*k | 5 |
| div.d f6, f1, f2 | f6 <= v1[i] / v2[i] | 5 |
| add.d f7, f6, f5 | f7 <= v3[i]\*k + v1[i] / v2[i] | 2 |
| s.d f7,v4(r1) | v4[i] <= f7 | 1 |
| daddui r1,r1,8 | r1 <= r1 + 8 | 1 |
| daddi r2,r2,-1 | r2 <= r2 - 1 | 1 |
| bnez r2,loop |  | 2 |
| halt |  | 1 |
| total |  | 216 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| daddui r1,r0,0 | f | d | x | m | W | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| daddui r2,r0,10 |  | f | d | x | m | W | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| l.d f1,v1(r1) |  |  | f | d | x | m | w | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| l.d f2,v2(r1) |  |  |  | f | d | x | m | W | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| l.d f3,v3(r1) |  |  |  |  | f | d | x | m | w | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| mul.d f5,f3,f4 |  |  |  |  |  | f | d | raw | Xm1 | Xm2 | Xm3 | Xm4 | m | W | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| div.d f6, f1, f2 |  |  |  |  |  |  | f |  | d | Xd | xd | xd | xd | xd | xd | xd | xd | m | w | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| add.d f7, f6, f5 |  |  |  |  |  |  |  |  | f | d | raw | raw | raw | raw | raw | raw | raw | Xf1 | Xf2 | m | w | 2 |  |  |  |  |  |  |  |  |  |  |  |  |
| s.d f7,v4(r1) |  |  |  |  |  |  |  |  |  | f |  |  |  |  |  |  |  | d | x | s | m | W | 1 |  |  |  |  |  |  |  |  |  |  |  |
| daddui r1,r1,8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | f | d |  | x | m | W | 1 |  |  |  |  |  |  |  |  |  |  |
| daddi r2,r2,-1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | f |  | d | x | m | W | 1 |  |  |  |  |  |  |  |  |  |
| bnez r2,loop |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | f | raw | d | x | m | W | 2 |  |  |  |  |  |  |  |
| halt |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | f |  |  |  | 1 |  |  |  |  |  |  |  |

6+210=216