情報工学実験3 LSI班 最終課題

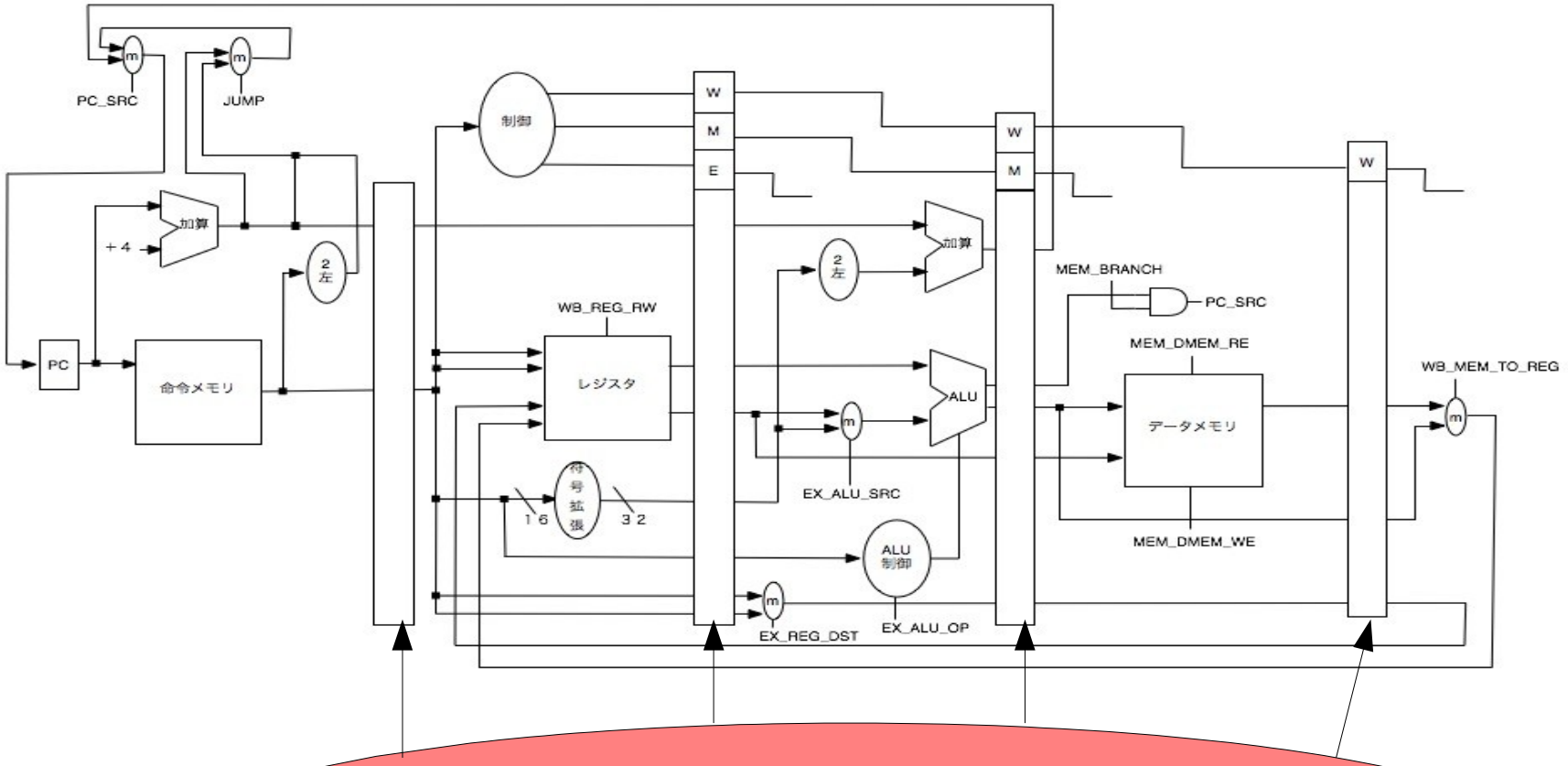
Mini-MIPSのパイプライン化とハザード回避
-分岐予測を用いたハザードフリー-

055755C：真玉橋朝明

発表内容

- Mini-MIPS のパイプライン化
- 分岐予測を用いた制御ハザードの回避

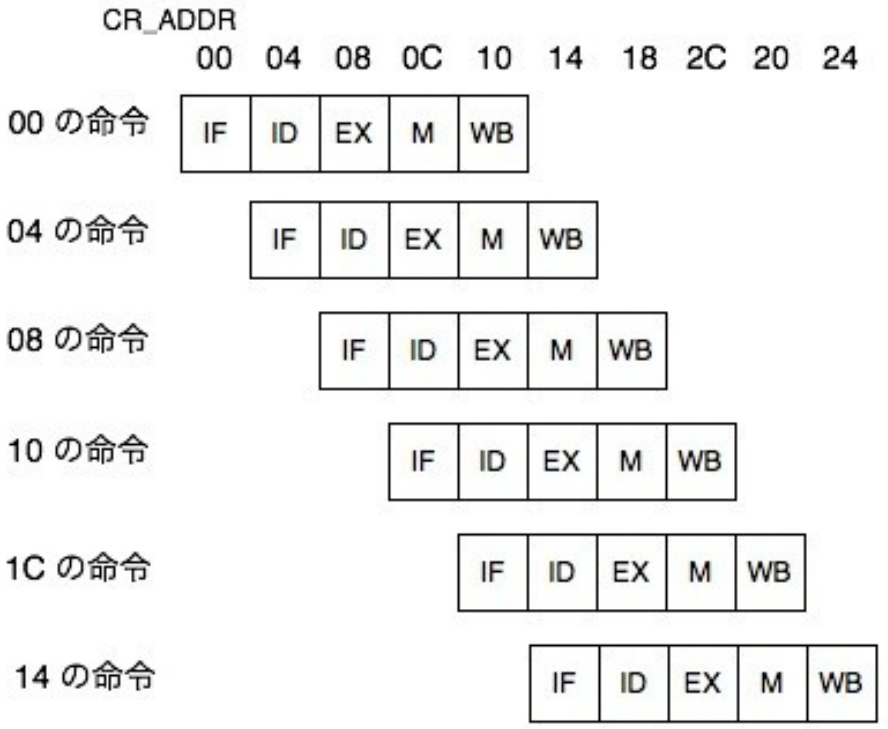
パイプライン化したMini-MIPS のブロック図



変更点：
・ステージを5つにわける
・各ステージ間にレジスタを設置

パイプラインの動作とVHDL記述による実現

```
--IF/ID
  process (CLK, RESET) begin
    if ( RESET = '1' ) then
      ID_INC_ADDR      <= (others => '0');
      ID_BR_ADDR       <= (others => '0');
      ID_INST          <= (others => '0');
      ID_NJUMP         <= '0';
      ID_BR_PREDICTION <= (others => '0');
      ID_PRE_WADDR     <= (others => '0');
    elsif ( CLK'event and CLK = '1' ) then
      ID_INC_ADDR      <= INC_ADDR;
      ID_BR_ADDR       <= BR_ADDR;
      ID_INST          <= INST;
      ID_NJUMP         <= NJUMP;
      ID_BR_PREDICTION <= BR_PREDICTION;
      ID_PRE_WADDR     <= PRE_WADDR
    end if;
```



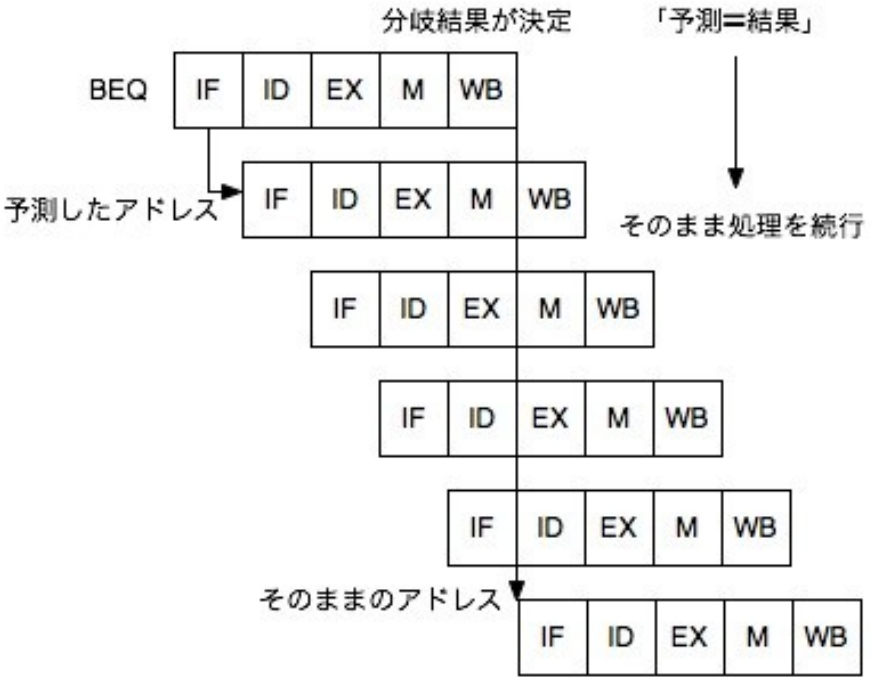
分岐予測を用いたハザード回避

分岐予測とは・・これまでの分岐の結果に基づき分岐が起こるかどうかを予測して処理を進める方法

実現に必要な機能・・

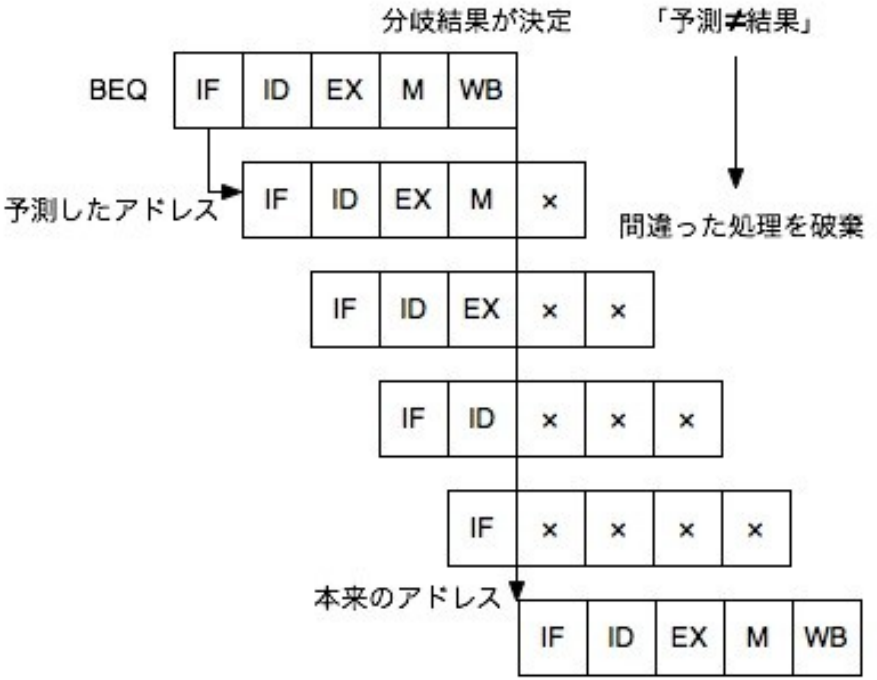
- 分岐結果の履歴を保存するメモリ
- 履歴のメモリを参照して分岐の予測をおこなう制御
- JUMP 命令にたいして無条件に分岐する制御
- 予測を外したときに修正を行う制御

処理の流れ



「分岐しない」と予測　=> 実際にも「分岐しない」

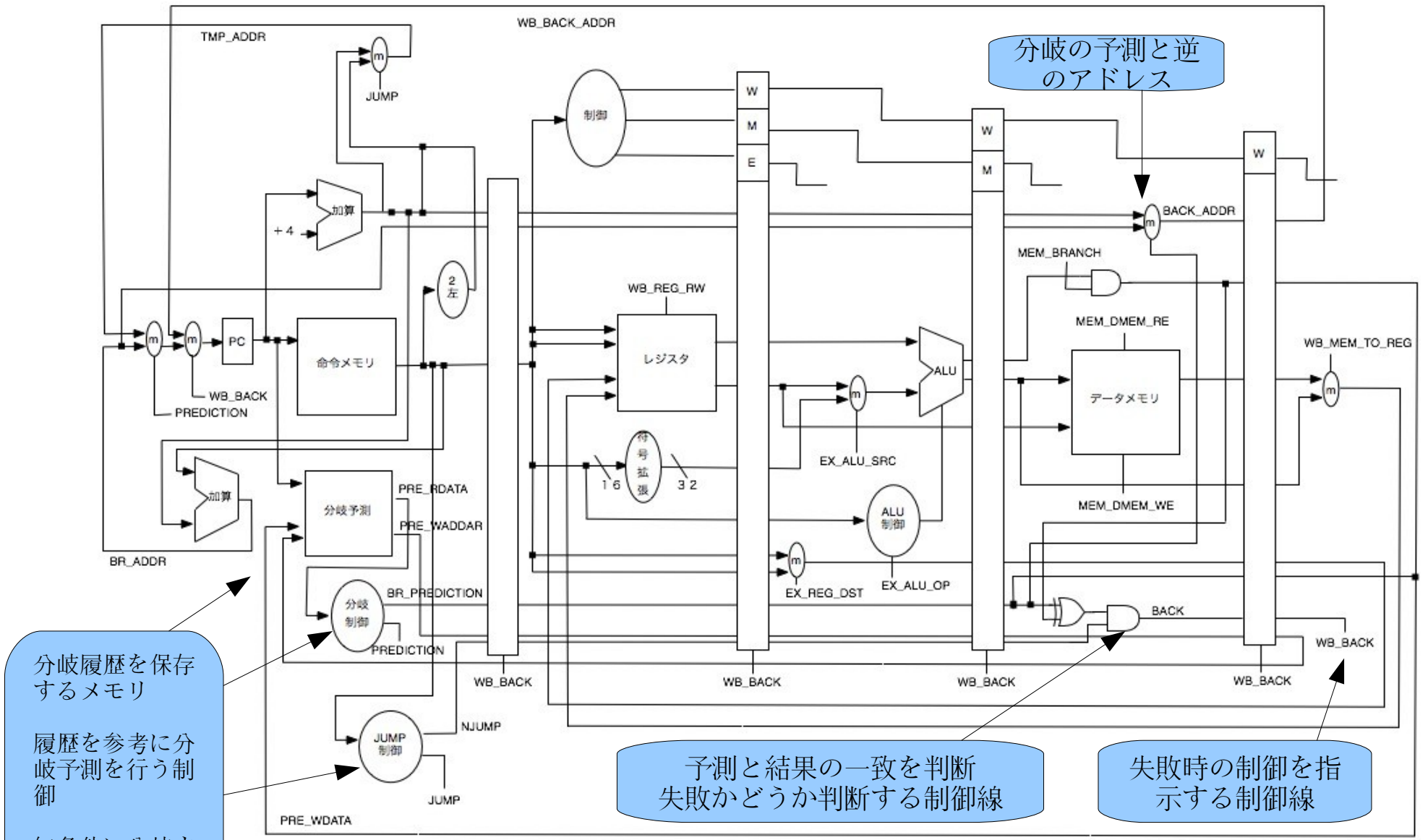
「分岐する」と予測　=> 実際にも「分岐する」



「分岐しない」と予測　=> 実際は「分岐する」

「分岐する」　と予測　=> 実際は「分岐しない」

分岐予測を実装した回路のブロック図



分岐履歴を保存するメモリ

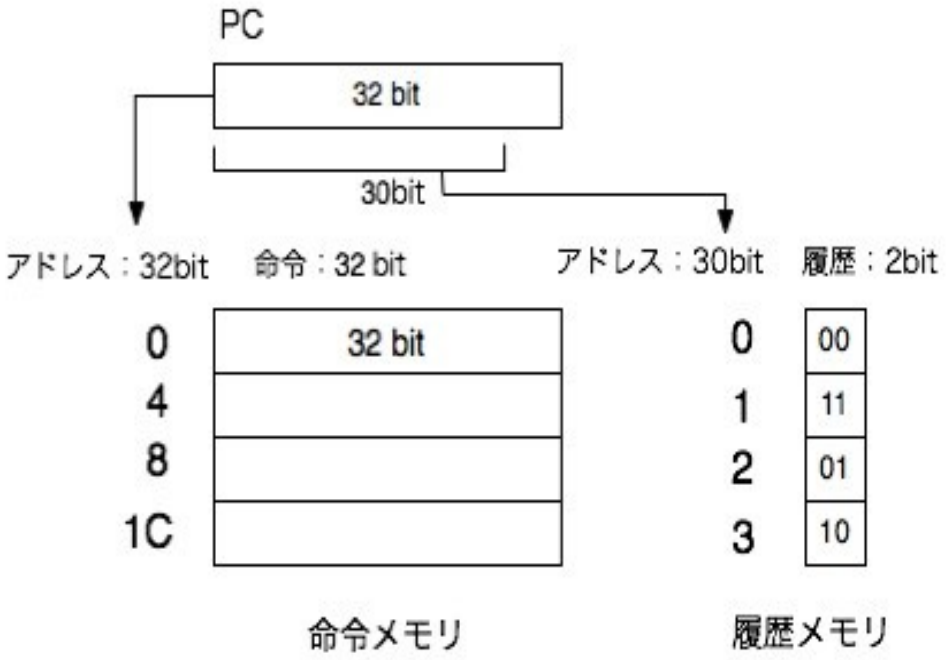
履歴を参考に分岐予測を行う制御

無条件に分岐する制御

予測と結果の一致を判断
失敗かどうか判断する制御線

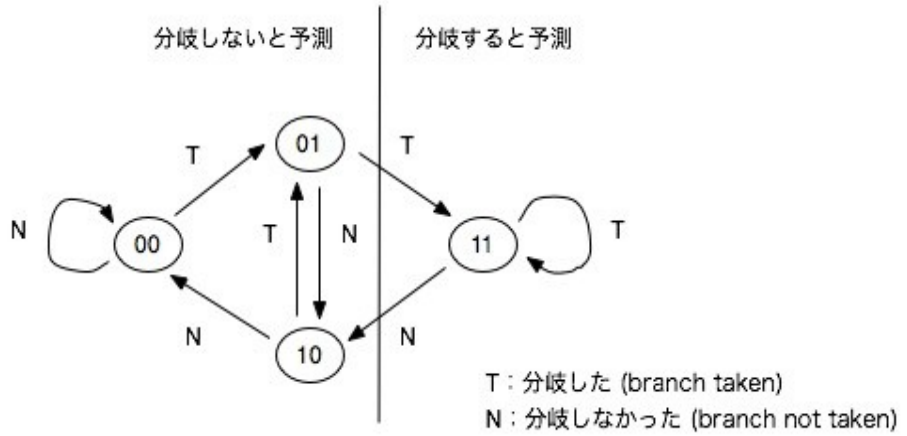
失敗時の制御を指示する制御線

履歴のメモリと制御



・履歴のメモリ

プログラムカウンタに読み込まれた命令アドレスから分岐の履歴を読み出す．前回と前々回の２bitのデータが保持されている



・分岐を予測する

履歴のデータをもとに分岐を予測する．

失敗時の消去

```
PC_SRC <= MEM_ALU_ZF and MEM_BRANCH;

RESULT <= PC_SRC xor MEM_BR_PREDICTION(1);

BACK   <= RESULT and MEM_NJUMP;
```

予測と結果が一致してるかどうかの確認
MEM_BR_PREDICTION(1)が予測．MEM_NJUMPはJUMP命令が出ていなとき '1' となる．

```
elsif ( CLK'event and CLK = '1' ) then
  if (WB_BACK = '1') then
    ID_INC_ADDR      <= ( others => '0');
    ID_BR_ADDR       <= ( others => '0');
    ID_INST          <= ( others => '0');
    ID_NJUMP         <= '0';
    ID_BR_PREDICTION <= ( others => '0');
    ID_PRE_WADDR     <= ( others => '0');
  else
    ID_INC_ADDR      <= INC_ADDR;
    ID_BR_ADDR       <= BR_ADDR;
    ID_INST          <= INST;
    ID_NJUMP         <= NJUMP;
    ID_BR_PREDICTION <= BR_PREDICTION;
    ID_PRE_WADDR     <= PRE_WADDR;
  end if;
end if;
```

一致していなかったときの処理

パイプラインレジスタ内の処理に追加．クロックするときに
WB_BACK が '1'なら分岐命令以下の処理を破棄する．

シミュレーションによる確認

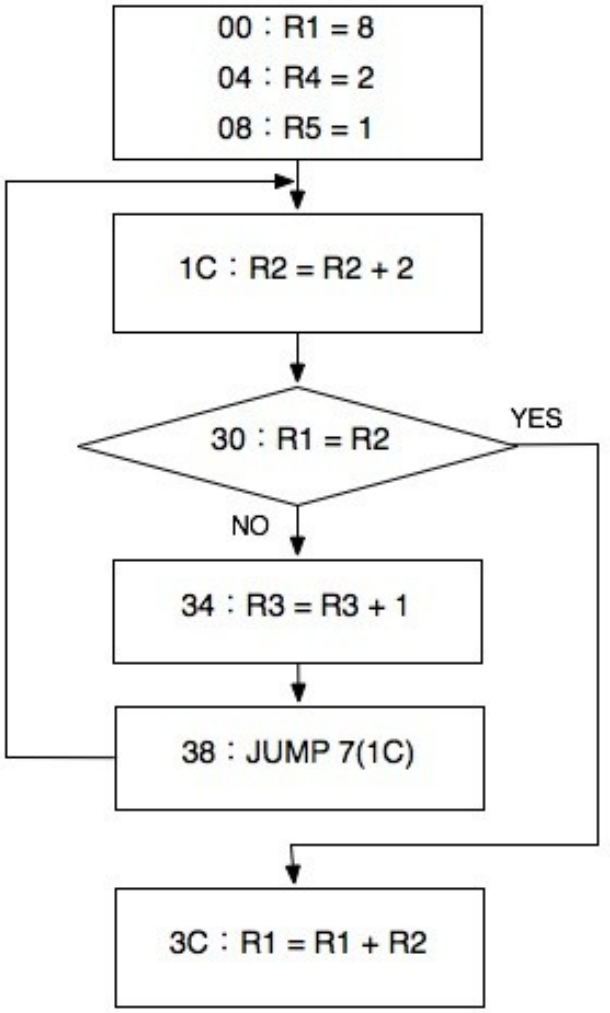
アセンブラプログラム

```
00 : ori  R1  R0  8
04 : ori  R4  R0  2
08 : ori  R5  R0  1
0C～18 : nop
1C : add  R2  R2 R4
20 ～ 2C : nop
30 : beq  R1  R2  2
34 : add  R3  R3 R5
38 : JUMP     7
3C : add  R1  R1 R2
```

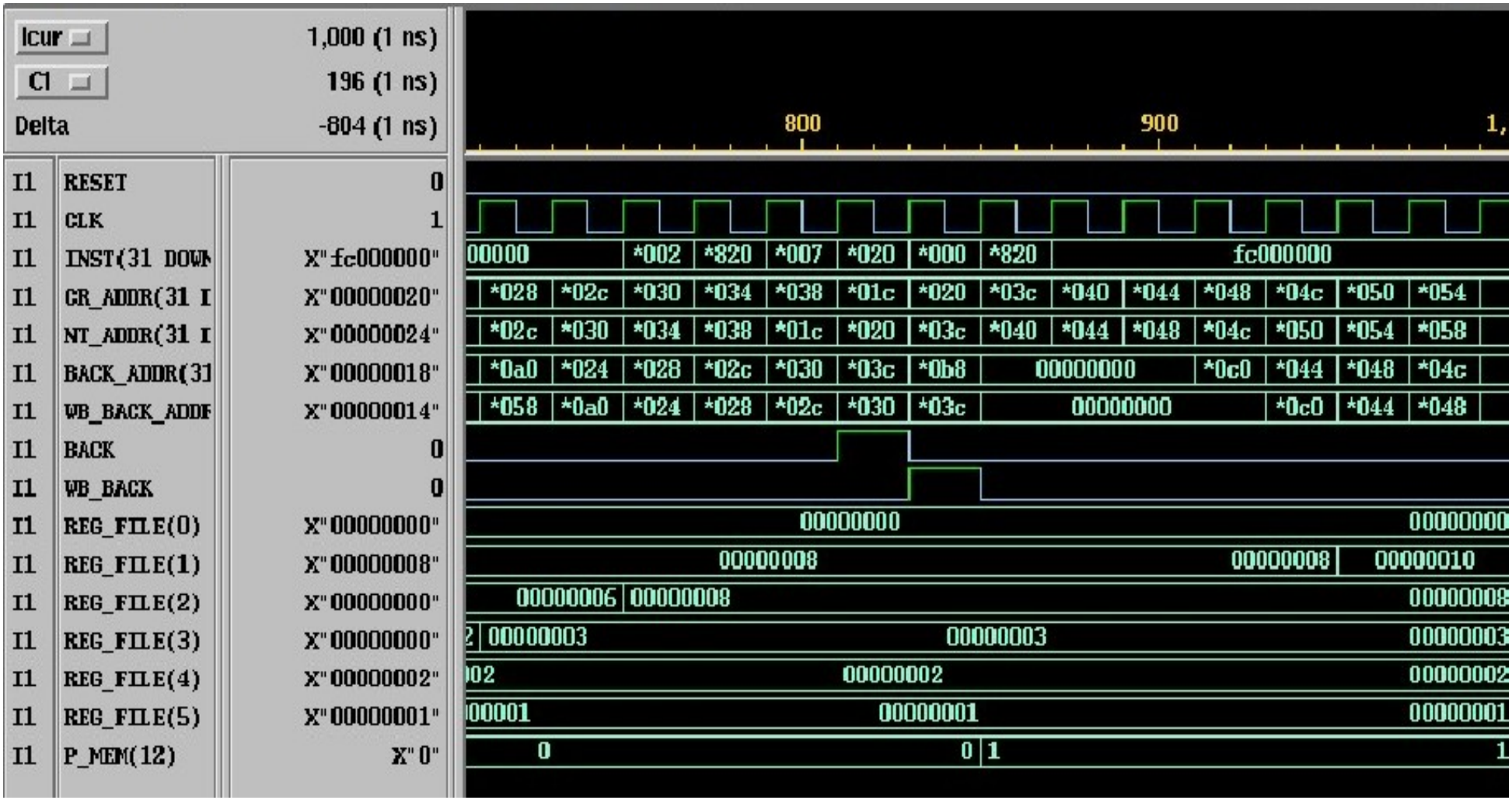
このプログラムが正しく動作すると各レジスタの値は以下のようになる.

R1= 16
R2= 8
R3= 3

プログラムの流れ



各レジスタの値と予測失敗時の信号 BACK = '1' のときの各値の変化を見る．
シミュレータの値は16進数なのでレジスタは
REG_FILE(1) =10, REG_FILE(2) = 8, REG_FILE(3) = 3 となる



BACK の WB ステージでの信号 WB_BACK が '1' となったとき，クロックすると CR_ADDR が分岐先のアドレスになってることがわかる．また，各レジスタもプログラム通りになっており，パイプラインプロセッサとして正常に動作しかつハザード回避できていることが確認できる．

工夫した点

- 命令アドレスが4番地ごとに対し，履歴のメモリアドレスを1番地ごとに変換することにより，メモリの容量を減らしたこと.

- 履歴のデータによる分岐予測を2回連続で分岐した場合のみ分岐することにしたこと.

工夫したい点

- 履歴のメモリが命令メモリと1対1で対応しているためデータの読み込みは速いが分岐命令以外の分のメモリもとっている．その部分のメモリをどうにかして容量を小さくしたい.

質疑応答