Fiscal Year 2021

Ver. 2022-01-06a

Course number: CSC.T433
School of Computing,
Graduate major in Computer Science

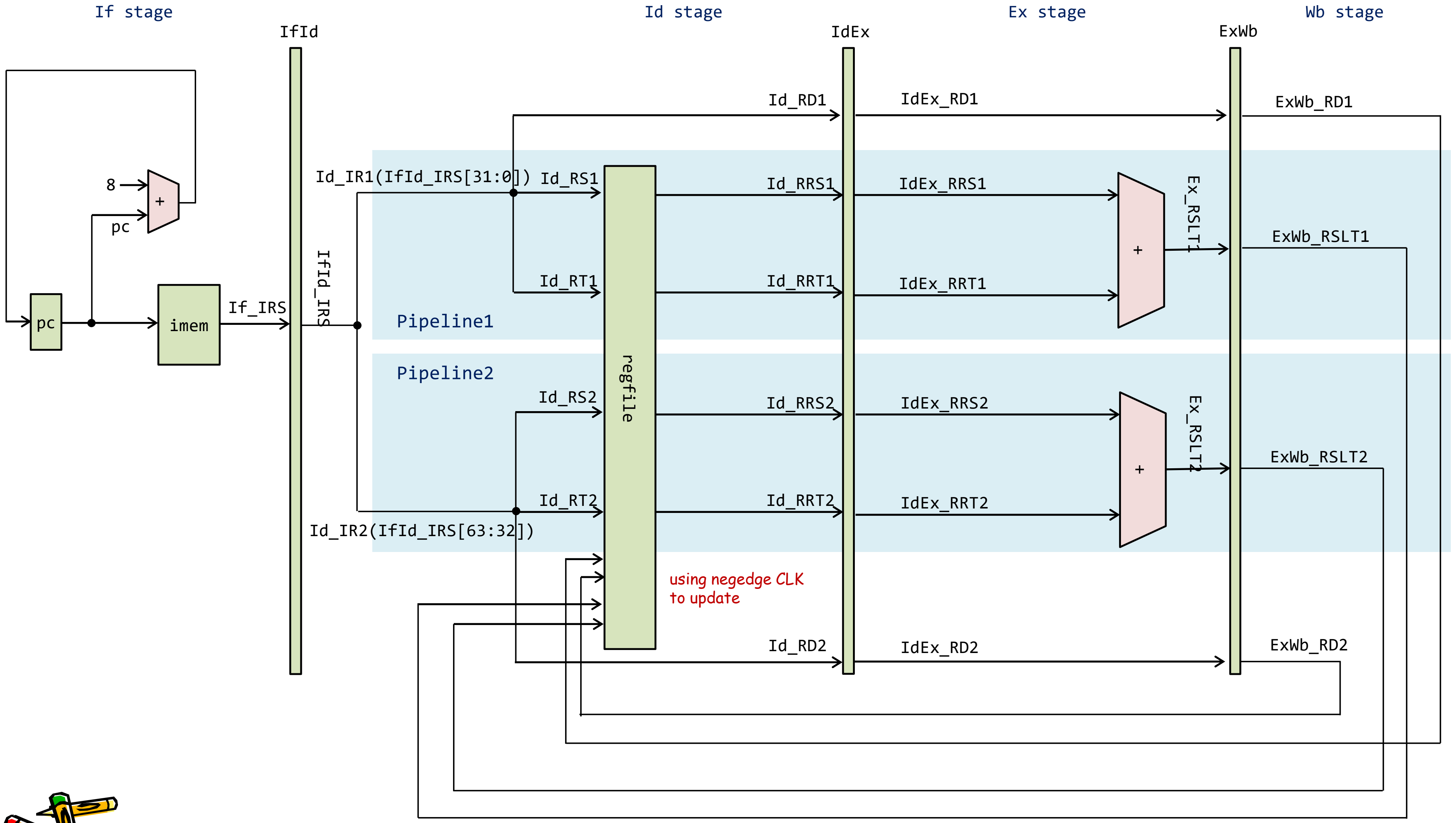
Advanced Computer Architecture

6. Instruction Level Parallelism: Instruction Fetch and Branch Prediction

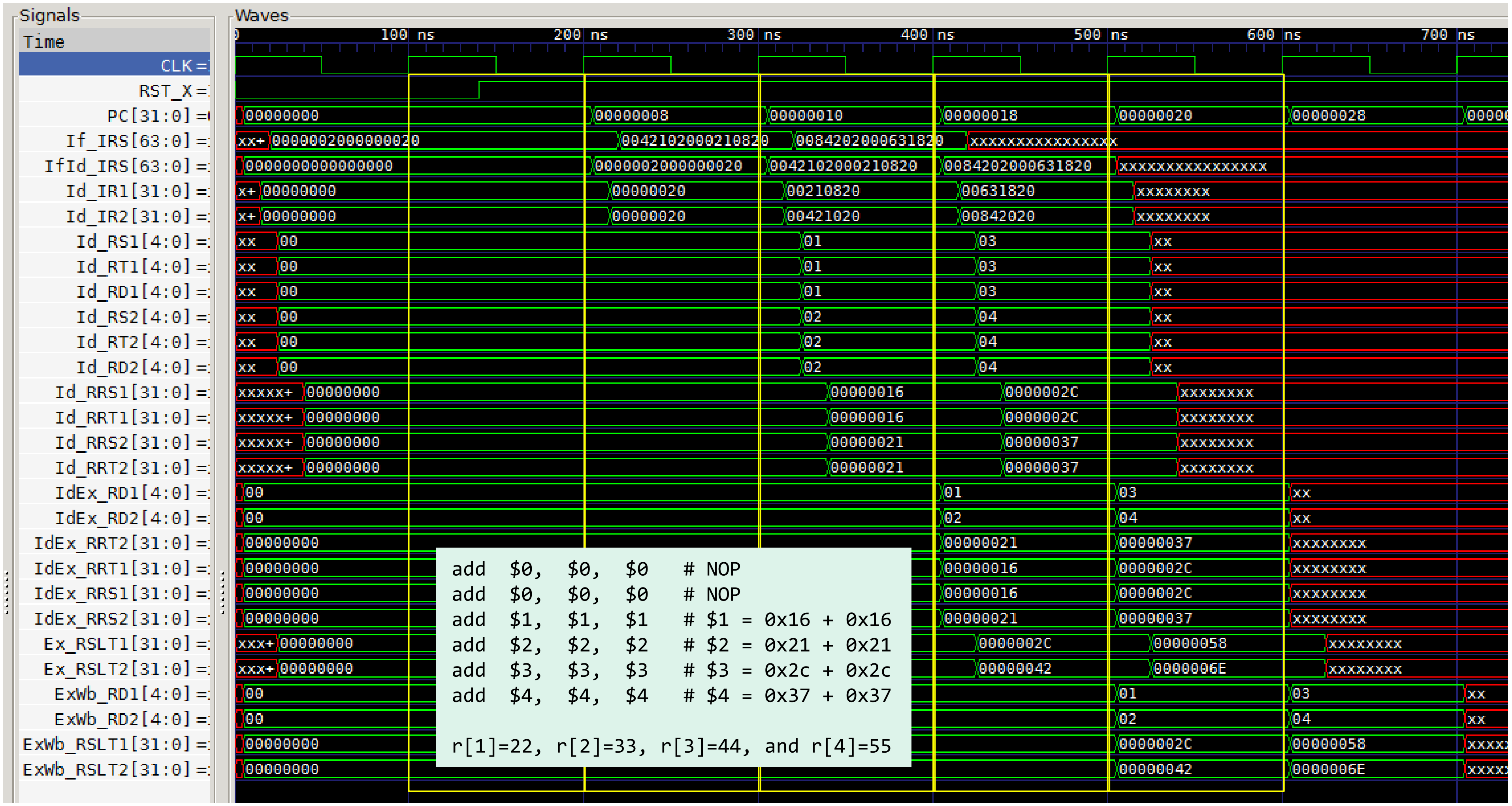
www.arch.cs.titech.ac.jp/lecture/ACA/
Room No.W936
Mon 14:20-16:00, Thr 14:20-16:00

Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp

A four stage pipelined 2-way superscalar processor supporting ADD which does not adopt data forwarding (proc10, Assignment 4)



Waveform of Proc10



Signals
Waves
Time 0 100 ns 200 ns 300 ns 400 ns 500 ns 600 ns 700 ns
CLK =
RST_X =
PC[31:0] = 00000000 00000008 00000010 00000018 00000020 00000028 0000
If_IRS[63:0] = xx+ 0000002000000020 0042102000210820 0084202000631820 xxxxxxxxxxxxxxxx
IfId_IRS[63:0] = 0000000000000000 0000002000000020 0042102000210820 0084202000631820 xxxxxxxxxxxxxxxx
Id_IR1[31:0] = x+ 00000000 00000020 00210820 00631820 xxxxxxxx
Id_IR2[31:0] = x+ 00000000 00000020 00421020 00842020 xxxxxxxx
Id_RS1[4:0] = xx 00 01 03 xx
Id_RT1[4:0] = xx 00 01 03 xx
Id_RD1[4:0] = xx 00 01 03 xx
Id_RS2[4:0] = xx 00 02 04 xx
Id_RT2[4:0] = xx 00 02 04 xx
Id_RD2[4:0] = xx 00 02 04 xx
Id_RRS1[31:0] = xxxxx+ 00000000 00000016 0000002C xxxxxxxx
Id_RRT1[31:0] = xxxxx+ 00000000 00000016 0000002C xxxxxxxx
Id_RRS2[31:0] = xxxxx+ 00000000 00000021 00000037 xxxxxxxx
Id_RRT2[31:0] = xxxxx+ 00000000 00000021 00000037 xxxxxxxx
IdEx_RD1[4:0] = 00 01 03 xx
IdEx_RD2[4:0] = 00 02 04 xx
IdEx_RRT2[31:0] = 00000000 00000021 00000037 xxxxxxxx
IdEx_RRT1[31:0] = 00000000 00000016 0000002C xxxxxxxx
IdEx_RRS1[31:0] = 00000000 00000016 0000002C xxxxxxxx
IdEx_RRS2[31:0] = 00000000 00000021 00000037 xxxxxxxx
Ex_RSLT1[31:0] = xxx+ 00000000 0000002C 00000058 xxxxxxxx
Ex_RSLT2[31:0] = xxx+ 00000000 00000042 0000006E xxxxxxxx
ExWb_RD1[4:0] = 00 01 03 xx
ExWb_RD2[4:0] = 00 02 04 xx
ExWb_RSLT1[31:0] = 00000000 0000002C 00000058 xxxx
ExWb_RSLT2[31:0] = 00000000 00000042 0000006E xxxx

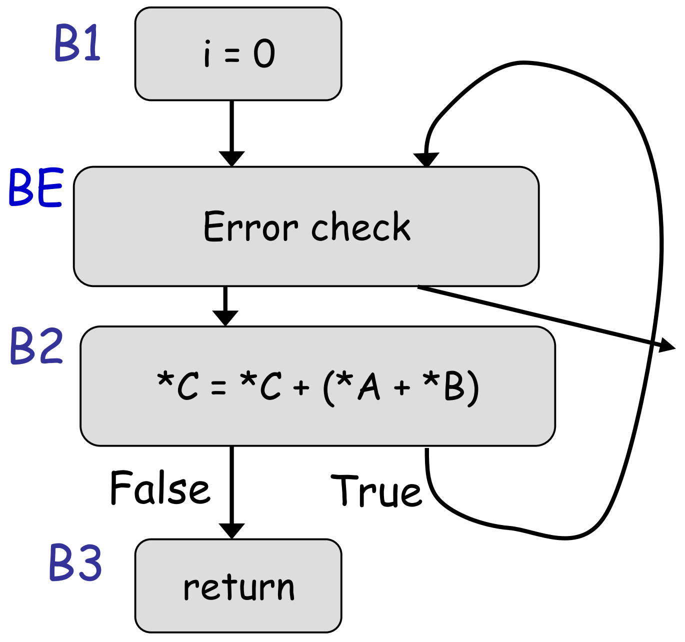
```
add  $0,  $0,  $0   # NOP
add  $0,  $0,  $0   # NOP
add  $1,  $1,  $1   # $1 = 0x16 + 0x16
add  $2,  $2,  $2   # $2 = 0x21 + 0x21
add  $3,  $3,  $3   # $3 = 0x2c + 0x2c
add  $4,  $4,  $4   # $4 = 0x37 + 0x37

r[1]=22, r[2]=33, r[3]=44, and r[4]=55
```

Branch predictor

- A branch predictor is a digital circuit that tries to guess or predict which way (taken or untaken) a branch will go before this is known definitively.
    - A random predictor will achieve about a 50% hit rate because the prediction output is 1 or 0.
    - Let's guess the accuracy. What is the accuracy of typical branch predictors for high-performance commercial processors?

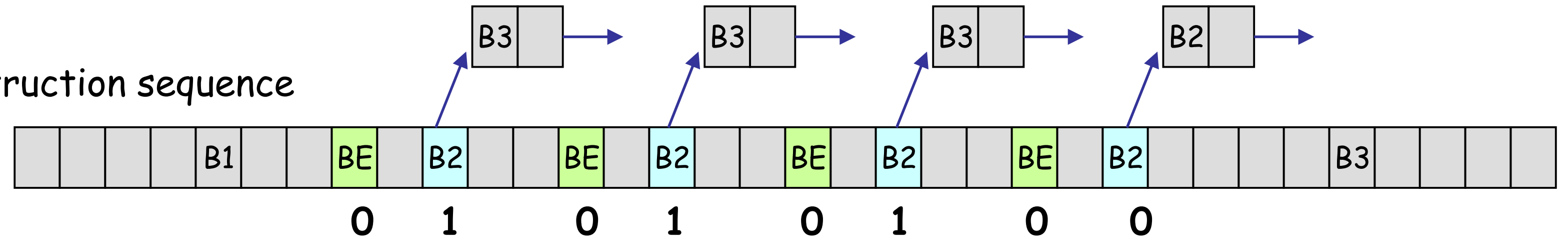
Sample program: vector add with two branches

```
#define VSIZE 4
void vadd(long *A, long *B, long *C){
  for(i=0; i<VSIZE; i++) {
    if(A[i]<0) error_routine();
    C[i] += (A[i] + B[i]);
  }
}
```



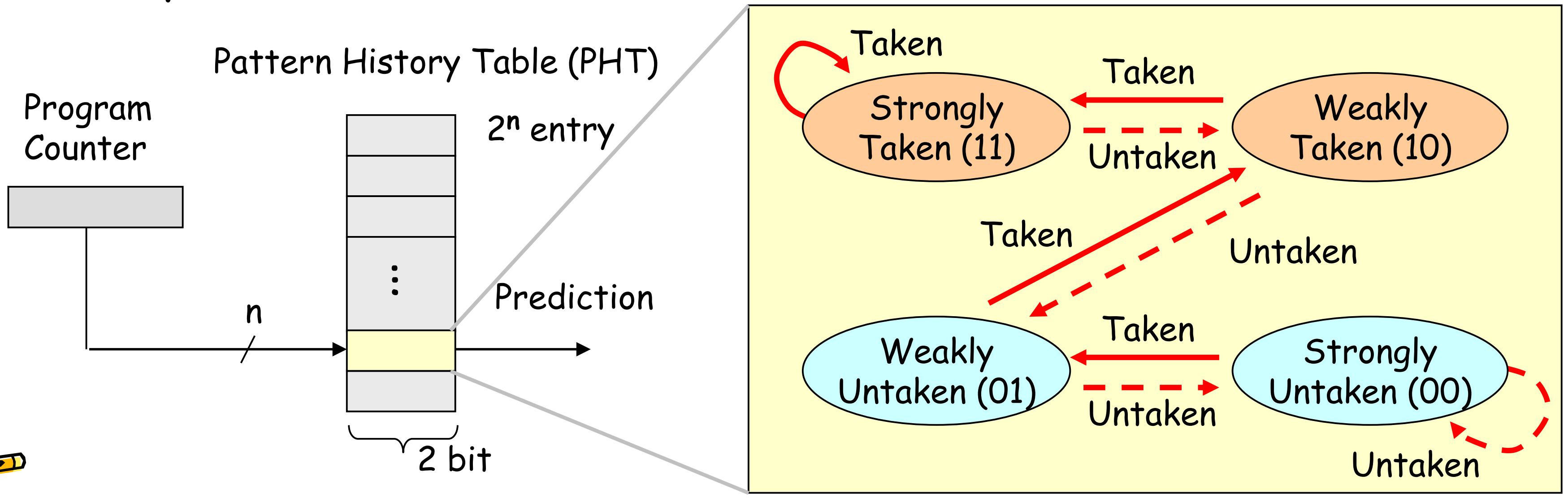
Control flow graph

Executed instruction sequence



Simple branch predictor: bimodal

- Program has many branch instructions. The behavior may depend on each branch. Use one counter for one branch instruction
- How to predict
  - Select one counter using PC, then it predicts 1 if the MSB of the register is one, otherwise predicts 0.
- How to update
  - Select one counter using PC, then update the counter in the same way as 2bit counter.



An innovation in branch predictors in 1993

- Using branch history
  - global branch history
  - local branch history
- 2-level branch predictor and Gshare

- Assume predicting the sequence 1110 1110 1110 1110 1110 …

```
11101110 ?
111011101 ?
1110111011 ?
11101110111 ?
111011101110 ?
```

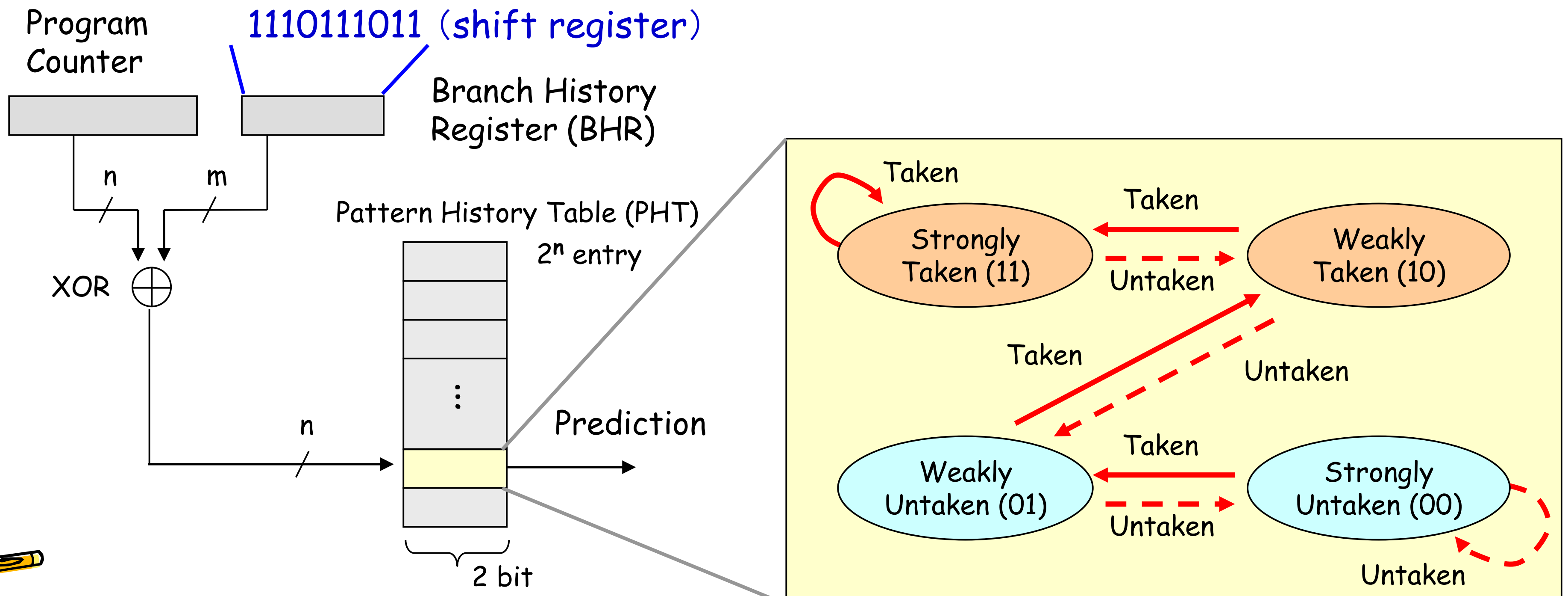
Recommended Reading

- Combining Branch Predictors
  - Scott McFarling, Digital Western Research Laboratory
  - WRL Technical Note TN-36, 1993

- A quote:
  "In this paper, we have presented two new methods for improving branch prediction performance. First, we showed that using the bit-wise exclusive OR of the global branch history and the branch address to access predictor counters results in better performance for a given counter array size."

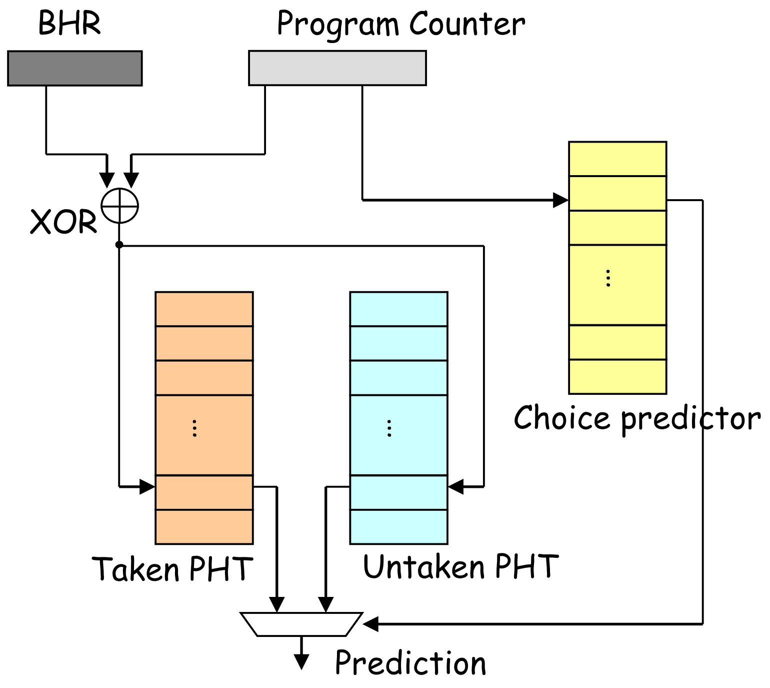
Gshare (TR-DEC 1993)

- How to predict
  - Using the exclusive OR of the global branch history and PC to access PHT, then MSB of the selected counter is the prediction.
- How to update
  - Shifting BHR one bit left and update LSB by branch outcome in IF stage.
  - Update the used counter in the same way as 2BC in WB stage.



Bi-Mode (MICRO 1997)

- A choice predictor (bimodal) is used as a meta-predictor
- How to predict
  - Like Gshare, both of Taken PHT and Untaken PHT make two predictions.
  - Select one among them by the choice predictor which tracks the global bias of a branch.
- How to update
  - The used PHT is updated in the same way as 2BC.
  - Choice predictor is update in the same way as bimodal



YAGS, Yet Another Global Scheme (MICRO 1998)

- Using two tagged PHTs
- When a PHT miss, choice PHT makes a prediction.

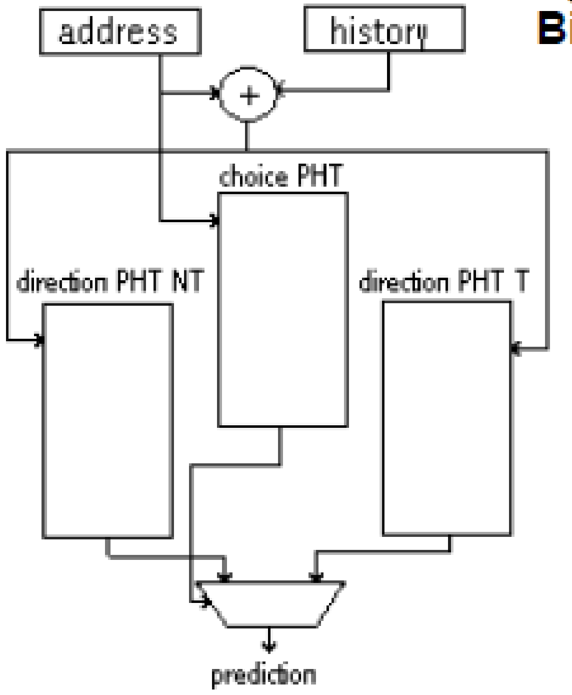


Figure 3. Bi-Mode

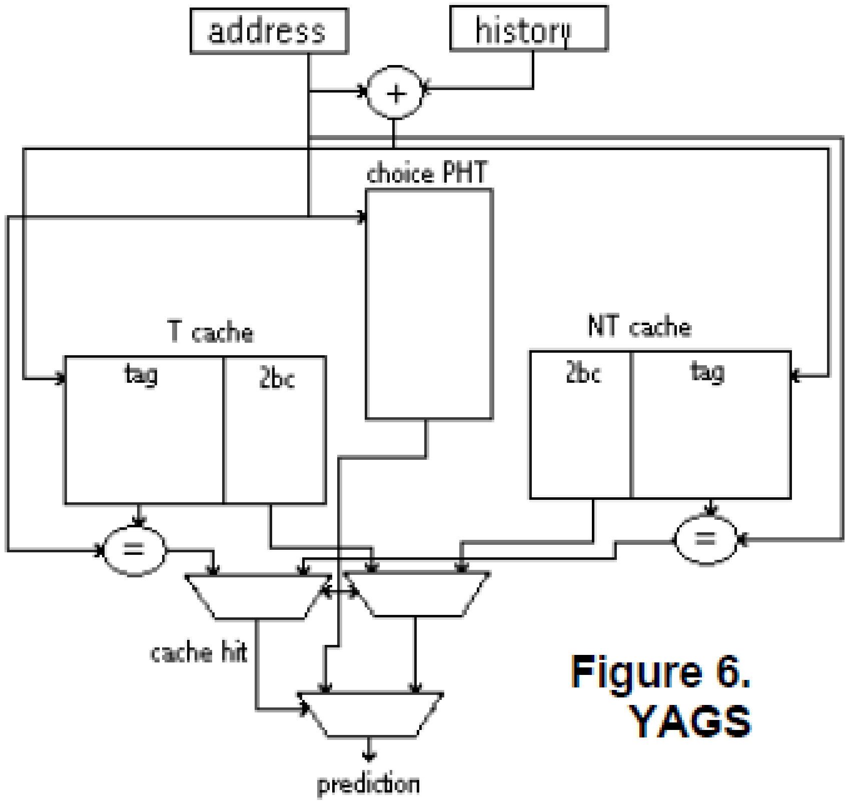


Figure 6. YAGS

From YAGS paper

An innovation in branch predictors in 1993 (again)

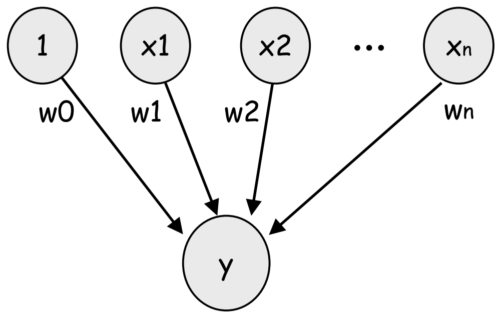
- Using branch history
  - global branch history
  - local branch history
- 2-level branch predictor and Gshare

- Assume predicting the sequence 1110 1110 1110 1110 1110 …

```
11101110 ?              11101110 ?
111011101 ?             111011101 ?
1110111011 ?            1110111011 ?
11101110111 ?           11101110111 ?
111011101110 ?          111011101110 ?
```

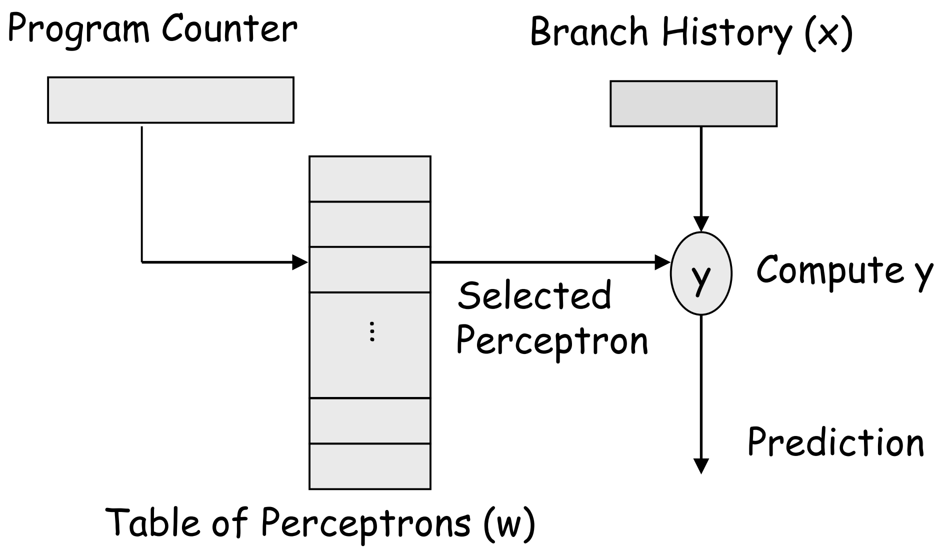
Perceptron (HPCA 2001)

- How to predict
  - Select one perceptron by PC
  - Compute y using the equation. It predicts 1 if y>=0, predicts 0 if y<0
- How to update
  - Train the weights of used perceptron when the prediction miss or |y| < T



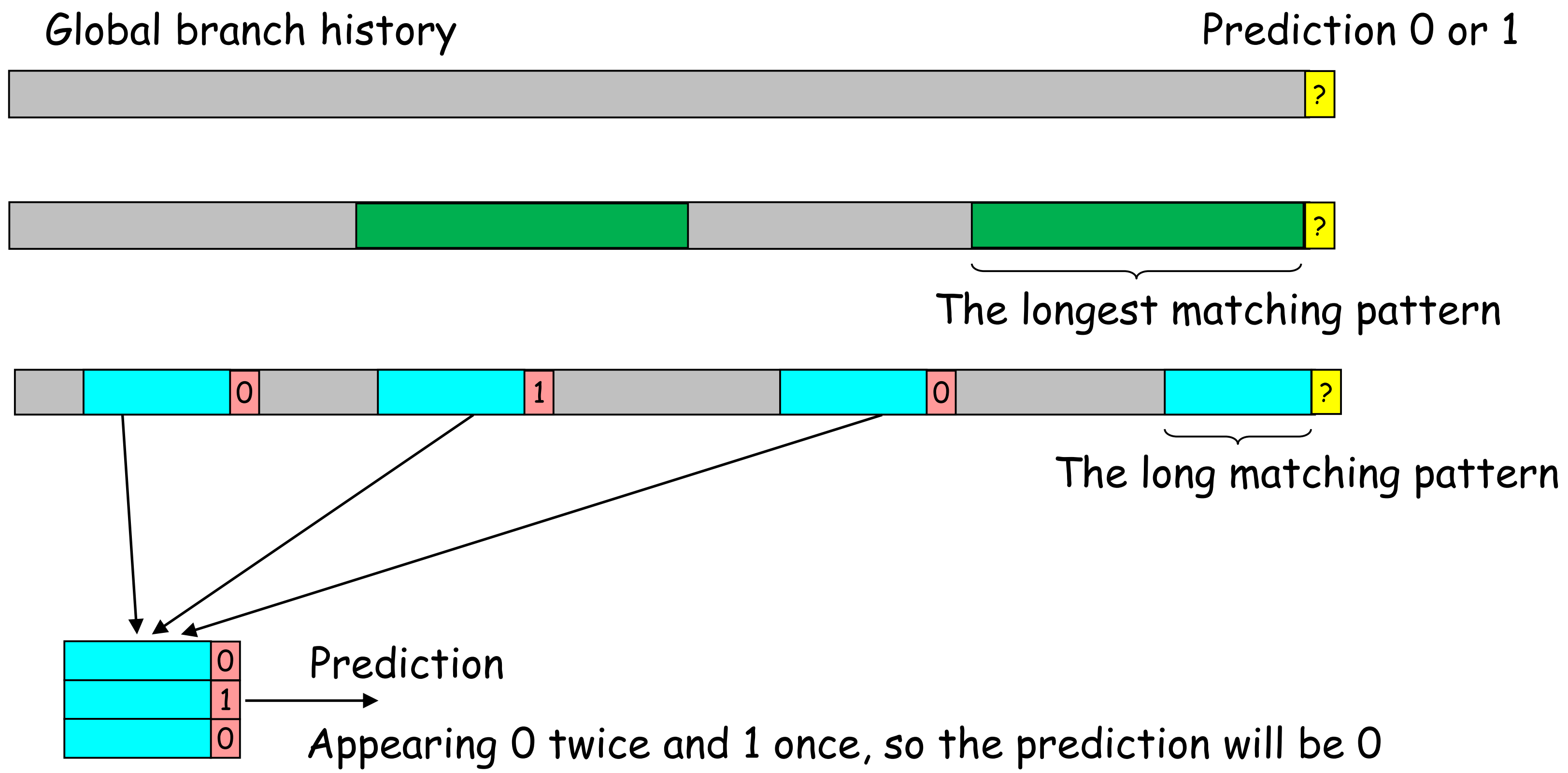
Perceptron Model

$$y = w_0 + \sum_{i=1}^{n} x_i w_i.$$

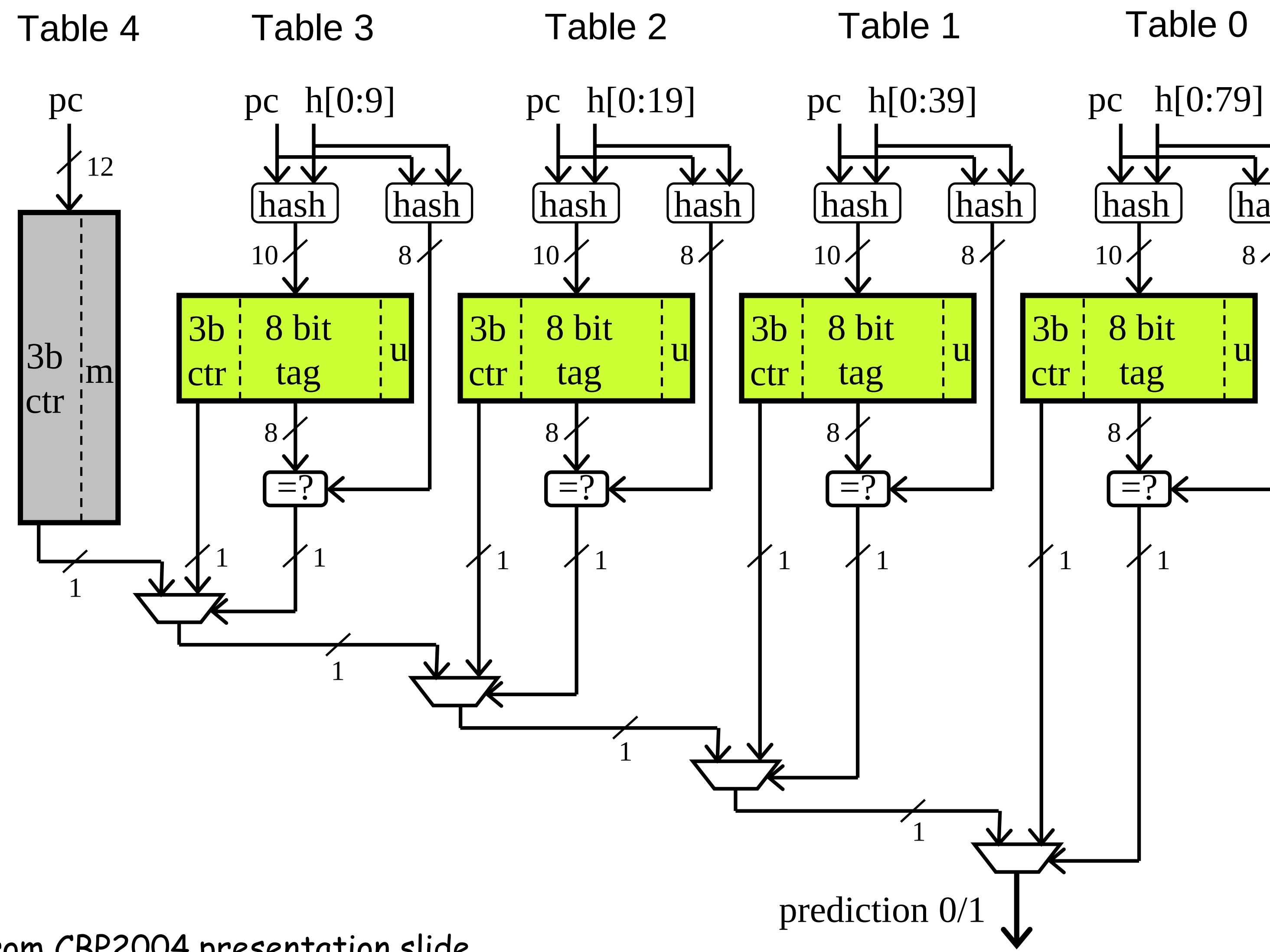


Branch predictors based on pattern matching

- Find the longest matching pattern (green rectangle)
- Select the proper matching length or long matching pattern (blue rectangle)
- Count the number of 0 and the number of 1 after the pattern (red rectangle), then predict.


Global branch history
Prediction 0 or 1
?
?
The longest matching pattern
0
1
0
?
The long matching pattern
0
1
0
Prediction
Appearing 0 twice and 1 once, so the prediction will be 0

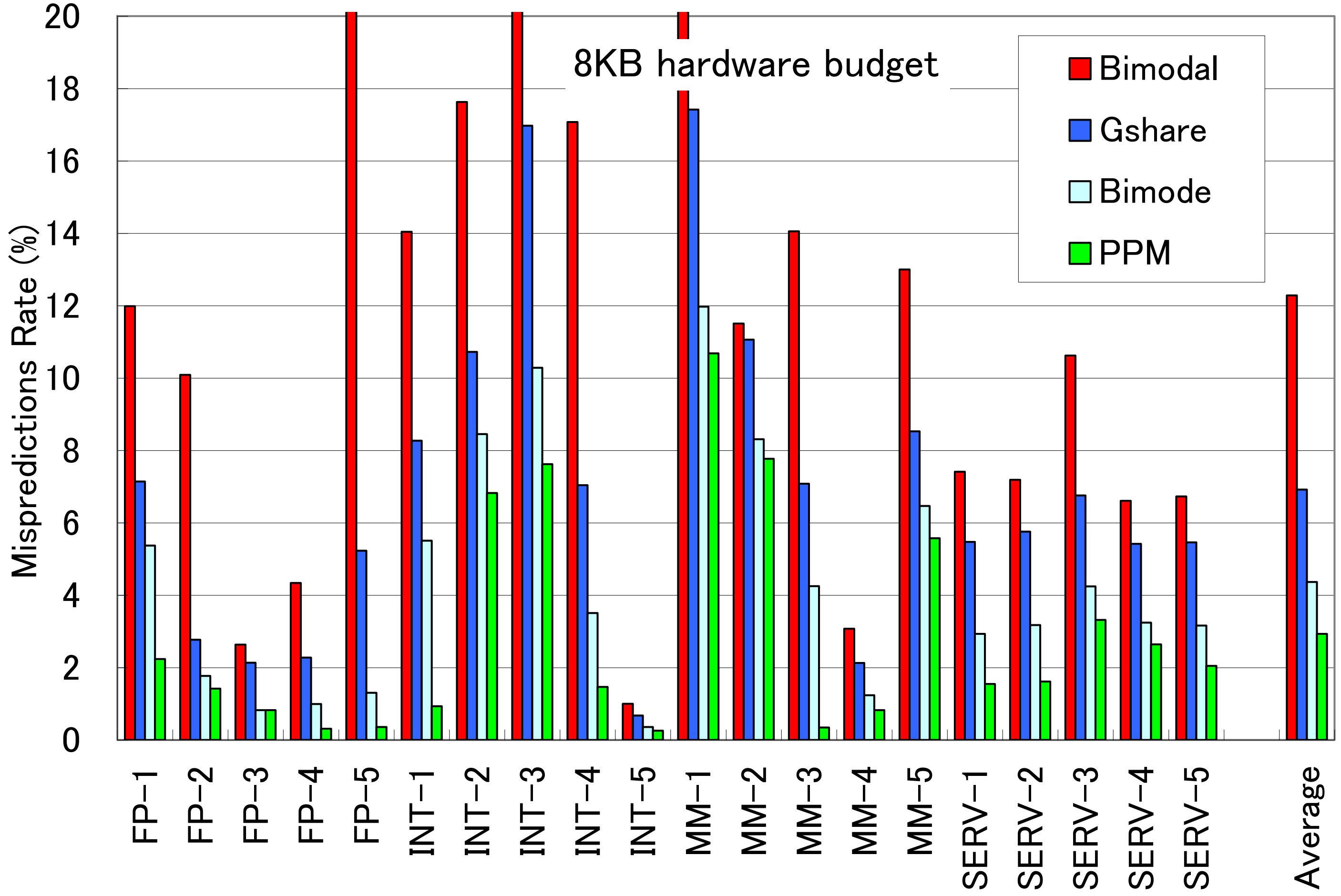
Partial Pattern Matching (CBP 2004)



From CBP2004 presentation slide

Prediction accuracy

- The accuracy of 4KB Gshare is about 93%.
- The accuracy of 4KB PPM is about 97%.



Recommended Reading

- Prophet-Critic Hybrid Branch Prediction
  - Ayose Falcon, UPC, Jared Stark, Intel, Alex Ramirez, UPC, Konrad Lai, Intel, Mateo Valero
  - ISCA-31 pp. 250-261 (2004)

A quote from Introduction (1/2)

Conventional predictors are analogous to a taxi with just one driver.

He gets the passenger to the destination using knowledge of the roads acquired from previous trips; i. e., using history information stored in the predictor's memory structures.

When he reaches an intersection, he uses this knowledge to decide which way to turn.

The driver accesses this knowledge in the context of his current location.

Modern branch predictors access it in the context of the current location (the program counter) plus a history of the most recent decisions that led to the current location.

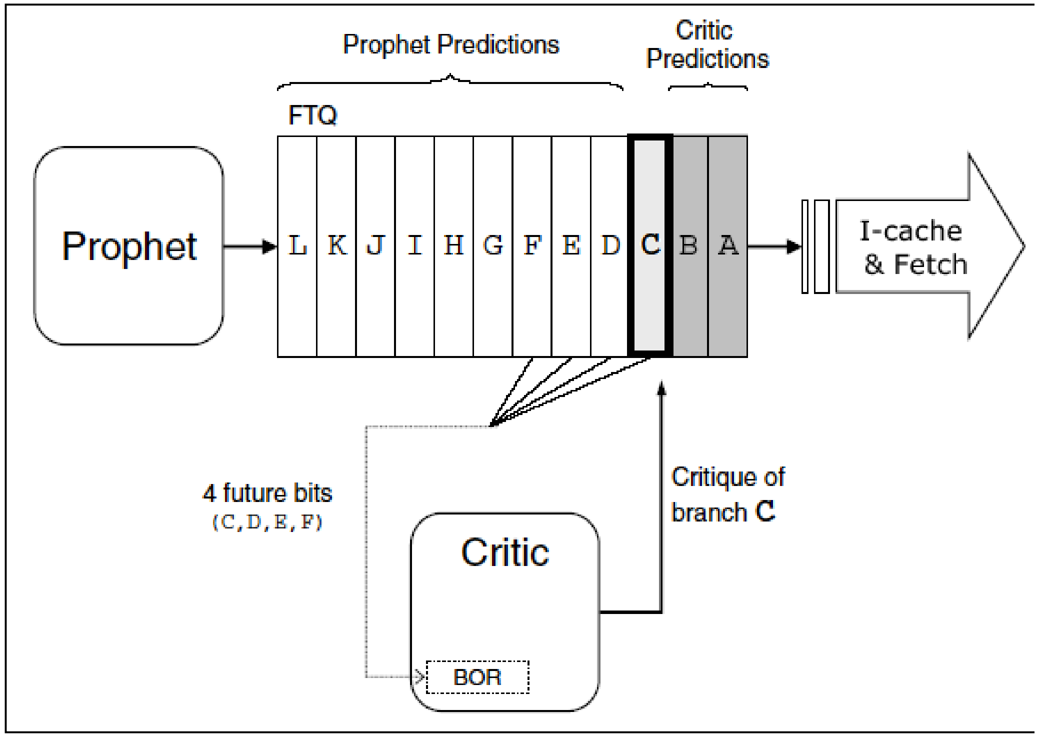
A quote from Introduction (2/2)

Prophet/critic hybrids are analogous to a taxi with two drivers: the front-seat and the back-seat. The front-seat driver has the same role as the driver in the single-driver taxi. This role is called the prophet.

The back-seat driver has the role of critic. She watches the turns the prophet makes at intersections. She doesn't say anything unless she thinks he's made a wrong turn. When she thinks he's made a wrong turn, she waits until he's made a few more turns to be certain they are lost. (Sometimes the prophet makes turns that initially look questionable, but, after he makes a few more turns, in hindsight appear to be correct.) Only when she's certain does she point out the mistake.

To recover, they backtrack to the intersection where she believes the wrong-turn was made and try a different direction.

Prophet-Critic Hybrid Branch Prediction



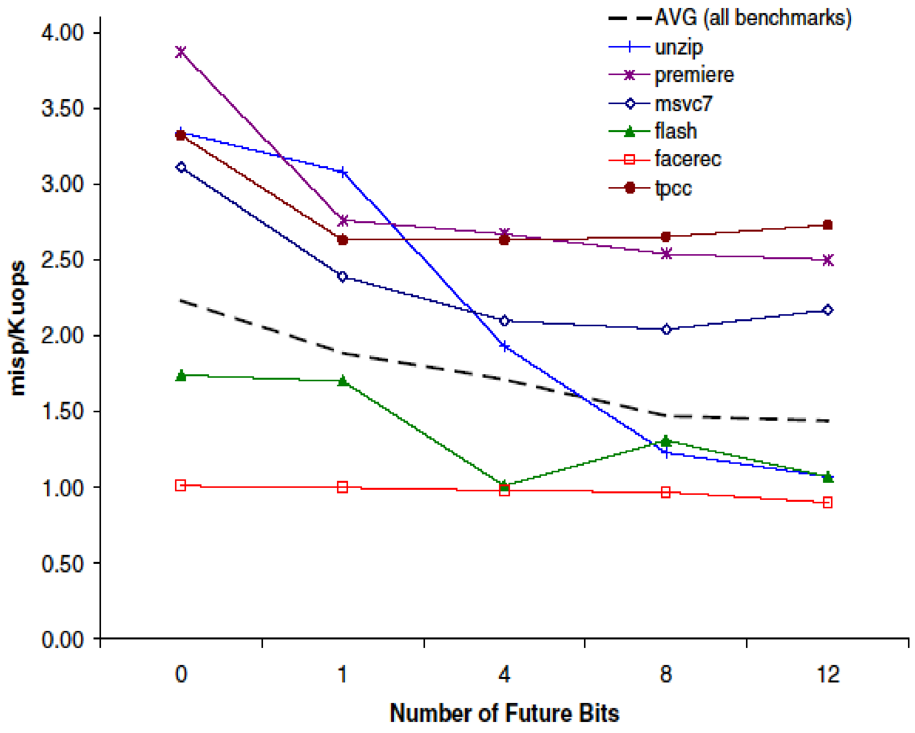


Figure 5. Effect of varying the number of future bits used by the critic on prediction accuracy for selected benchmarks. (prophet: 8KB perceptron; critic: 8KB tagged gshare)

Assignment 5

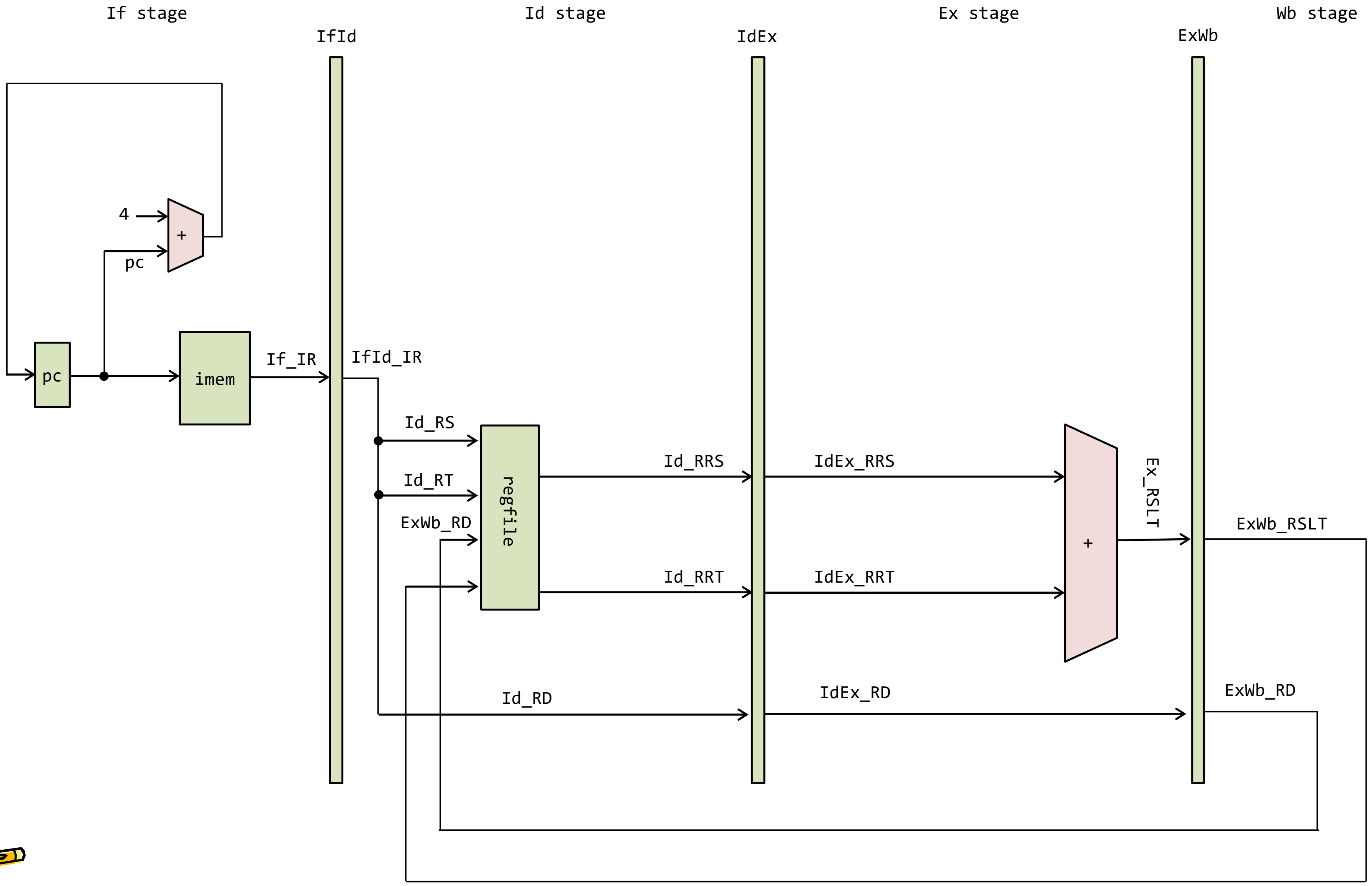
1. Design a four stage pipelined scalar processor supporting MIPS add and bne instruction in Verilog HDL. Please download proc06.v and proc07.v from the support page and refer it.
2. Verify the behavior of designed processor using following assembly code.

```
p.imem.mem[0] = {6'h0, 5'd0, 5'd0, 5'd0, 5'd0, 6'h20};  //     NOP
p.imem.mem[1] = {6'h0, 5'd5, 5'd1, 5'd5, 5'd0, 6'h20};  // L1: add  $5, $5, $1
p.imem.mem[2] = {6'h0, 5'd0, 5'd0, 5'd0, 5'd0, 6'h20};  //     NOP
p.imem.mem[3] = {6'h0, 5'd0, 5'd0, 5'd0, 5'd0, 6'h20};  //     NOP
p.imem.mem[4] = {6'h0, 5'd0, 5'd0, 5'd0, 5'd0, 6'h20};  //     NOP
p.imem.mem[5] = {6'h5, 5'd4, 5'd5, 16'hfffb};           //     bne  $4, $5, L1
p.imem.mem[6] = {6'h0, 5'd0, 5'd0, 5'd0, 5'd0, 6'h20};  //     NOP
p.imem.mem[7] = {6'h0, 5'd0, 5'd0, 5'd0, 5'd0, 6'h20};  //     NOP
p.imem.mem[8] = {6'h0, 5'd0, 5'd0, 5'd5, 5'd0, 6'h20};  //     add  $5, $0, $0
p.imem.mem[9] = {6'h0, 5'd0, 5'd0, 5'd0, 5'd0, 6'h20};  //     NOP
p.imem.mem[10]= {6'h0, 5'd0, 5'd0, 5'd0, 5'd0, 6'h20};  //     NOP
p.imem.mem[11]= {6'h5, 5'd2, 5'd0, 16'hfff5};           //     bne  $2, $0, L1
p.imem.mem[12]= {6'h0, 5'd0, 5'd0, 5'd0, 5'd0, 6'h20};  //     NOP
p.imem.mem[13]= {6'h0, 5'd0, 5'd0, 5'd0, 5'd0, 6'h20};  //     NOP
p.regfile.r[1] = 1; p.regfile.r[2] = 22; p.regfile.r[3] = 0; p.regfile.r[4] = 4; p.regfile.r[5] = 0;
```

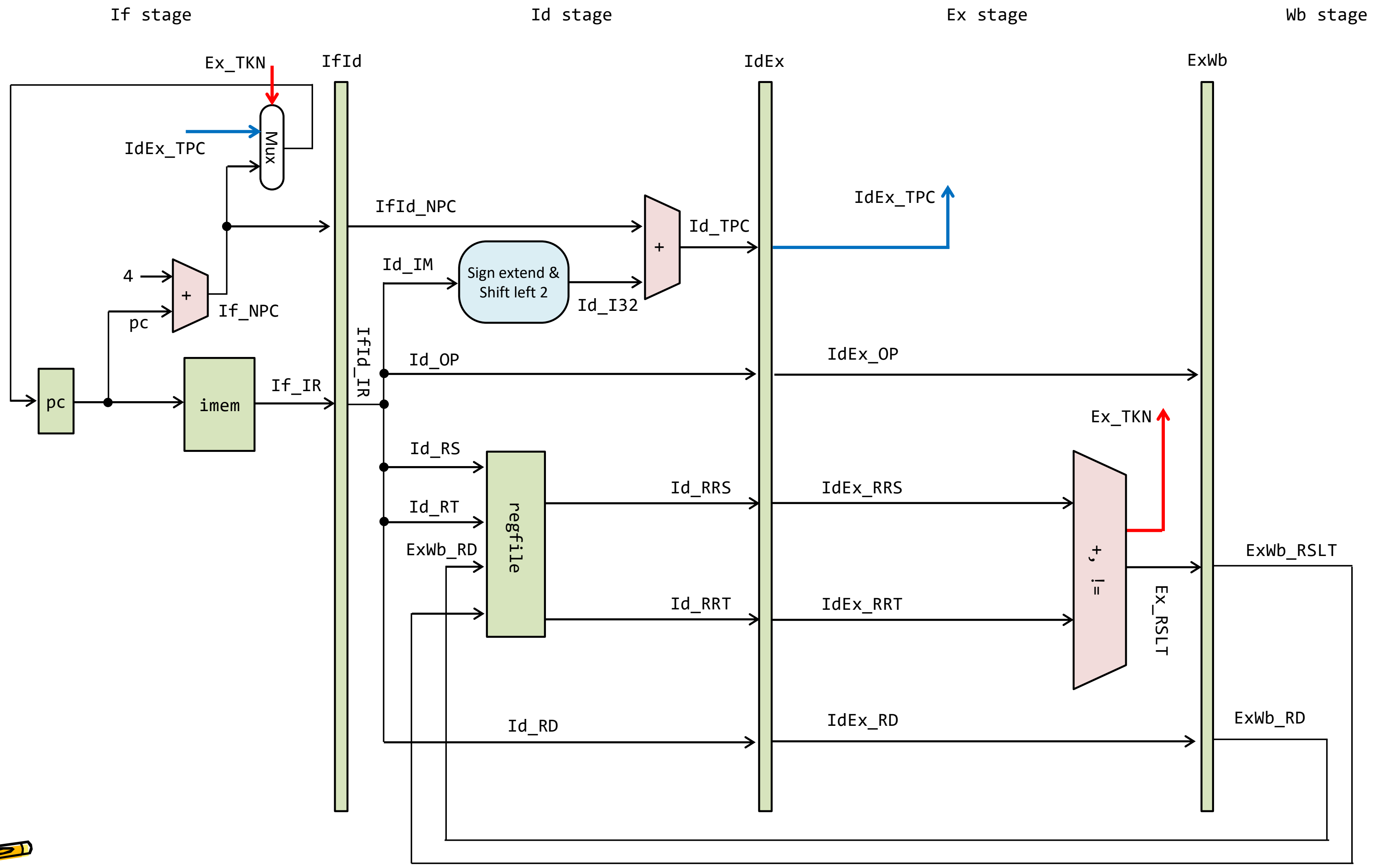
```
while(1){
  for(int i=1; i!=4; i++){

  }
}
```

3. Submit your report in a PDF file via E-mail **by the next Thursday**.
   - The report should include a block diagram, a source code in Verilog HDL, and obtained waveforms of your design.

Four stage pipelined processor supporting ADD, which does not adopt data forwarding (proc06.v, Assignment 3)



Four stage pipelined processor supporting ADD and BNE, which does not adopt data forwarding (proc08.v, Assignment 5)



Exercise: how to update PHT and BHR of Gshare