Fiscal Year 2019

Ver. 2019-12-18a

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

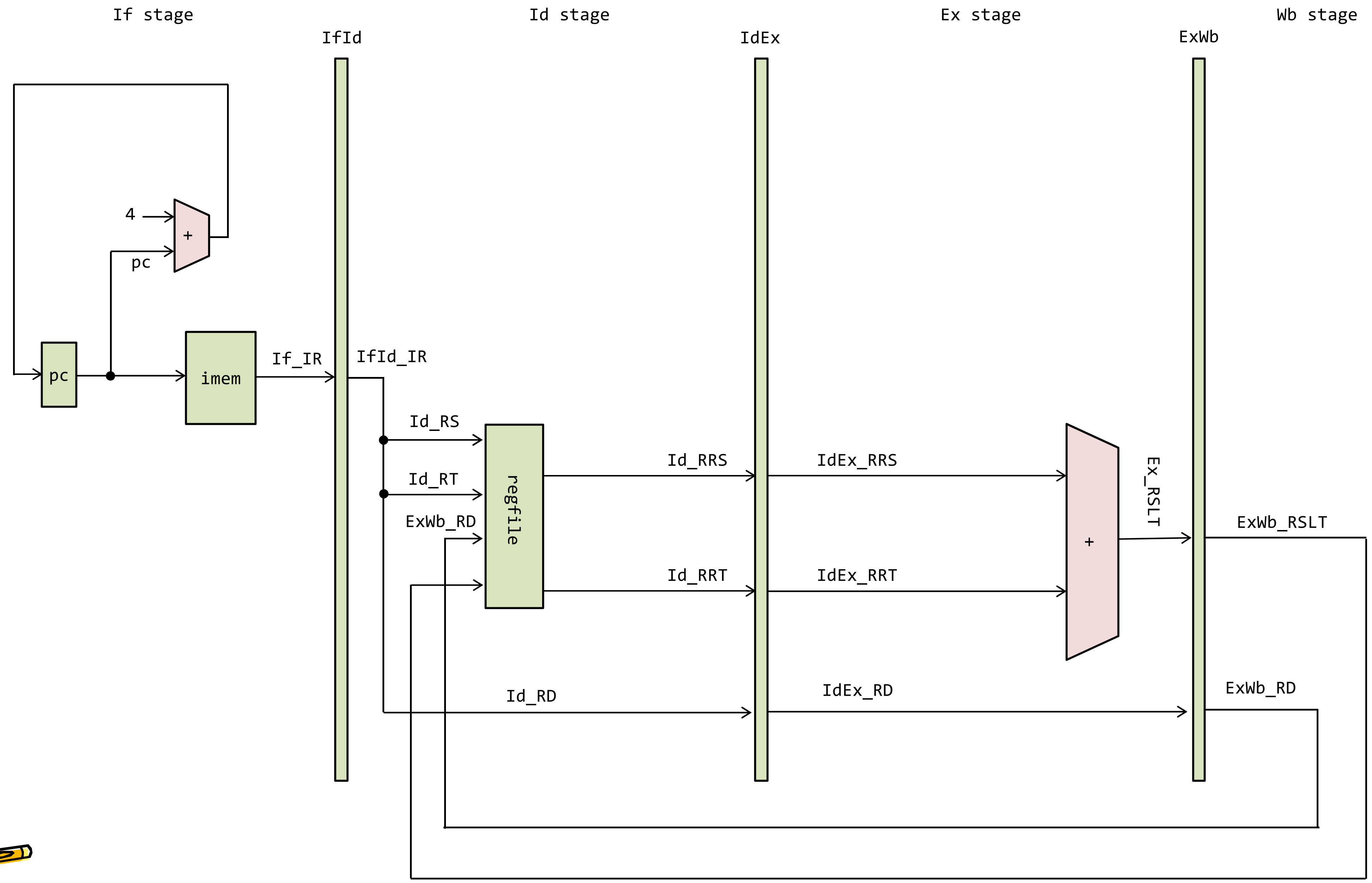
Advanced Computer Architecture

5. Instruction Level Parallelism: Concepts and Challenges

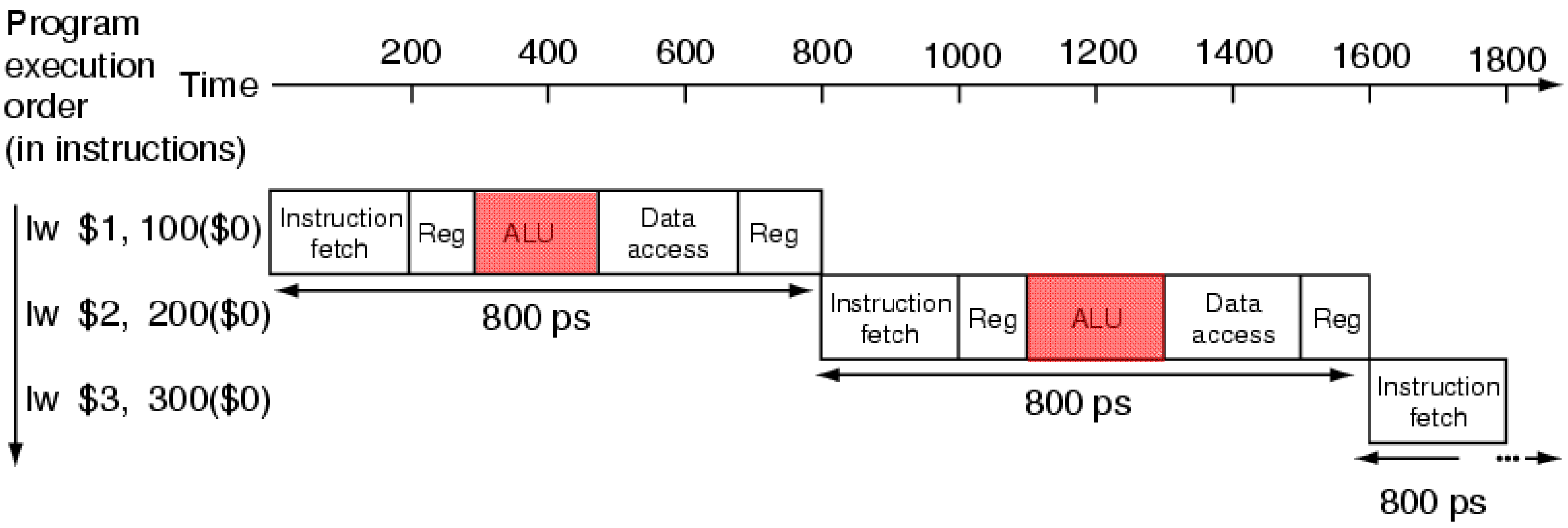
www.arch.cs.titech.ac.jp/lecture/ACA/
Room No.W936
Mon 13:20-14:50, Thr 13:20-14:50

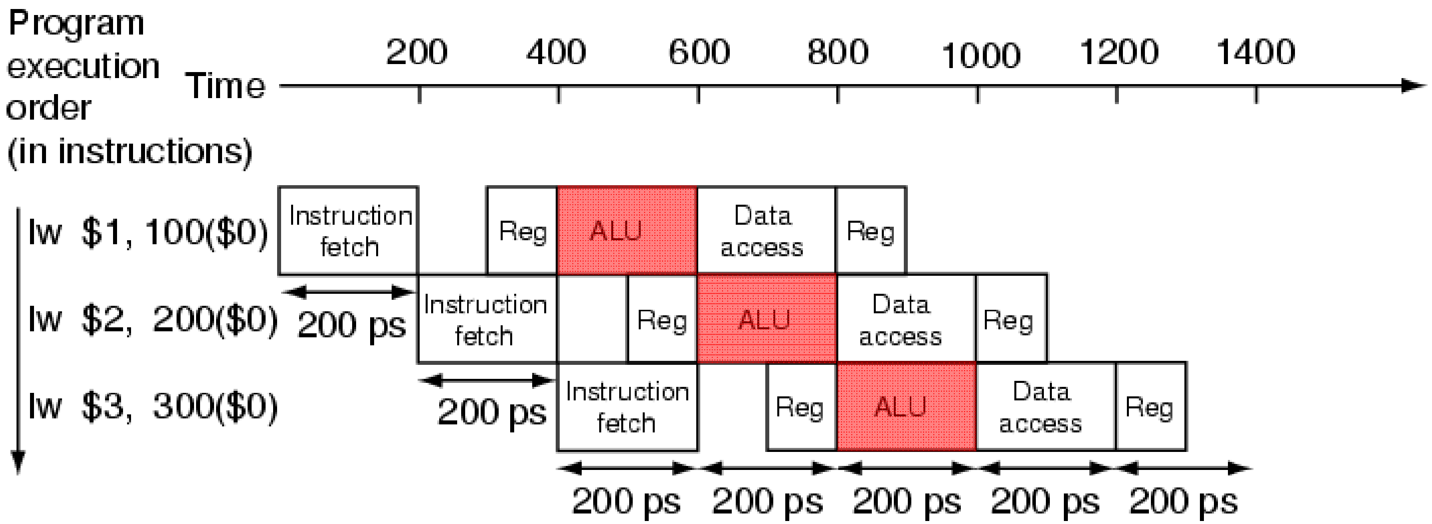
Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp

Four stage pipelined processor supporting ADD, which does not adopt data forwarding (proc06.v, Homework 4)



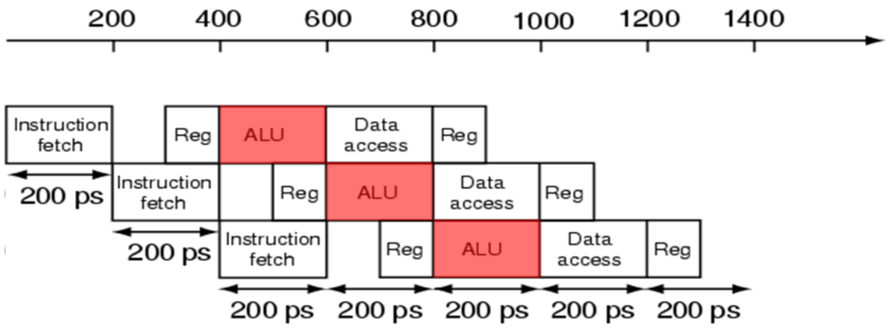
Single-cycle and pipelined processors



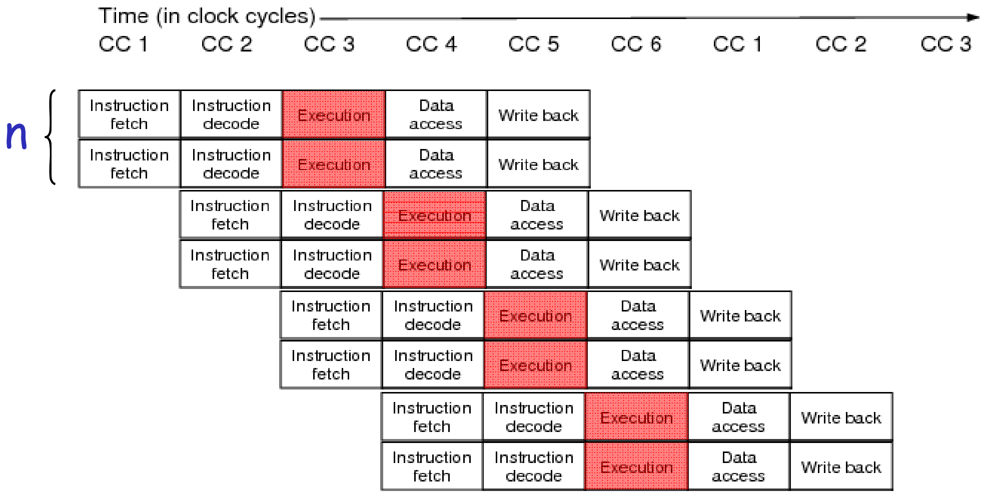


Scalar and Superscalar processors

- Scalar processor can execute at most one single instruction per clock cycle using one ALU.
  - IPC (Executed Instructions Per Cycle) is less than 1.
- Superscalar processor can execute more than one instruction per clock cycle by executing multiple instructions using multiple pipelines.
  - IPC (Executed Instructions Per Cycle) can be more than 1.
  - using n pipelines is called n-way superscalar



(a) pipeline diagram of scalar processor



(b) pipeline diagram of 2-way superscalar processor

Exercise: datapath of a 2-way superscalar

- Datapath of a 2-way superscalar processor supporting ADD, which does not adopt data forwarding

Assignment 5

1. Design a four stage pipelined 2-way superscalar processor supporting MIPS add instruction in Verilog HDL. Please download proc06.v from the support page and refer it.
2. Verify the behavior of designed processor using following assembly code
   assuming initial values of r[1]=22, r[2]=33, r[3]=44, and r[4]=55
   - `add  $0,  $0,  $0   #`
   - `add  $0,  $0,  $0   #`
   - `add  $1,  $1,  $1   #`
   - `add  $2,  $2,  $2   #`
   - `add  $3,  $3,  $3   #`
   - `add  $4,  $4,  $4   #`
3. Submit a report printed on A4 paper at the beginning of the next lecture.
   - The report should include a block diagram, a source code in Verilog HDL, and obtained waveforms of your design.

Exploiting Instruction Level parallelism (ILP)

- A superscalar processor has to handle some flows efficiently to exploit ILP
  - Control flow
    - To execute *n* instructions per clock cycle, the processor has to fetch at least *n* instructions per cycle.
    - The main obstacles are branch instruction (BNE, BEQ)
    - Another obstacle is instruction cache
  - Register data flow
  - Memory data flow

MIPS Control Flow Instructions

- MIPS conditional branch instructions:

```
bne $s0, $s1, Lbl   # go to Lbl if $s0≠$s1
beq $s0, $s1, Lbl   # go to Lbl if $s0=$s1
```

  - Ex: `if (i==j) h = i + j;`

```
    bne $s0, $s1, Lbl1
    add $s3, $s0, $s1
Lbl1:  ...
```

- Instruction Format (I format):

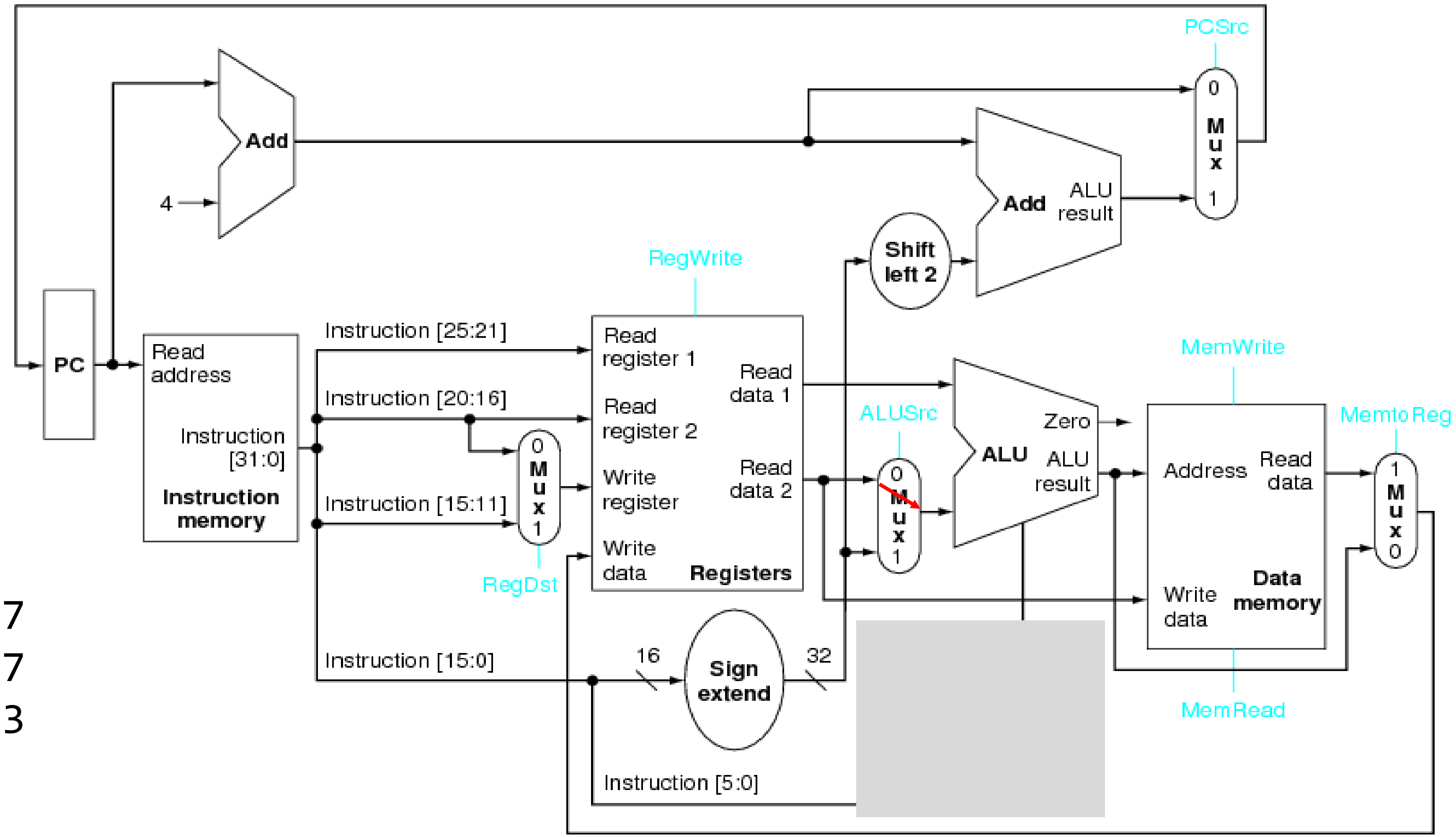
op	rs	rt	16 bit offset

- How is the branch destination address specified?

Datapath of processor supporting ADD, ADDI, LW, SW, BNE, BEQ

	IR[25:21]	IR[20:16]		
op	rs	rt	16 bit immediate	I format

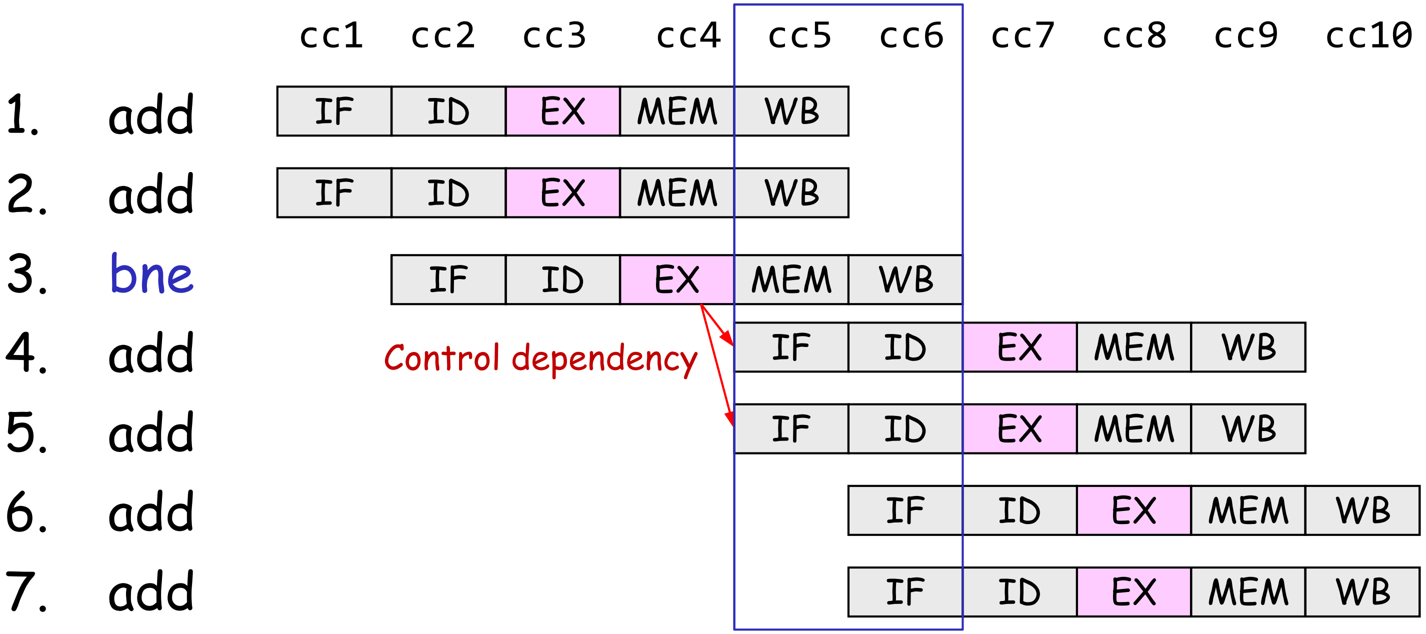
```
0x810  beq $t0, $t1, Lb      [ beq $8, $9, Lb ]
```



```
$8  =  7
$9  =  7
imm = -3
```

Why do branch instructions degrade IPC?

- The branch taken / untaken is determined in execution stage of the branch.
- The conservative approach of stalling instruction fetch until the branch direction is determined.



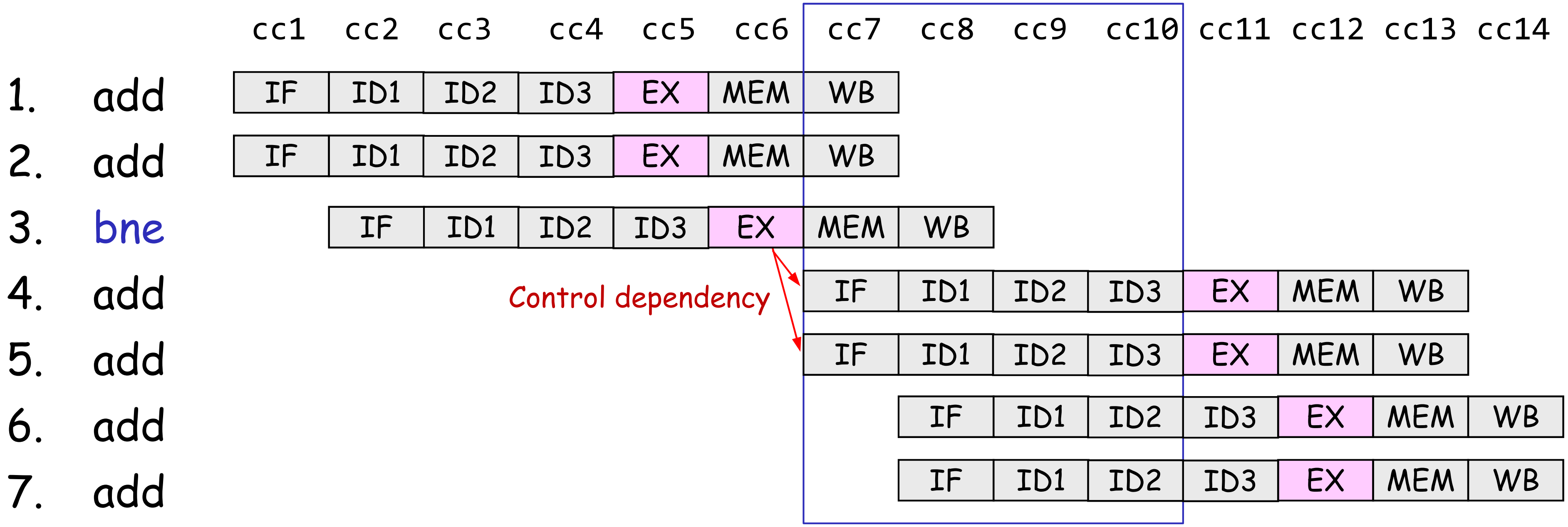
		cc1	cc2	cc3	cc4	cc5	cc6	cc7	cc8	cc9	cc10
1.	add	IF	ID	EX	MEM	WB					
2.	add	IF	ID	EX	MEM	WB					
3.	bne		IF	ID	EX	MEM	WB				
4.	add					IF	ID	EX	MEM	WB	
5.	add					IF	ID	EX	MEM	WB	
6.	add						IF	ID	EX	MEM	WB
7.	add						IF	ID	EX	MEM	WB

2-way superscalar processor executing instruction sequence with a branch

Note that because of a branch instruction, only one instruction is executed in cc4 and no instructions are executed in CC6 and CC7. This reduces the IPS.

Deeper pipeline

- In conservative approach, IPC degradation will be significant by deeper pipeline



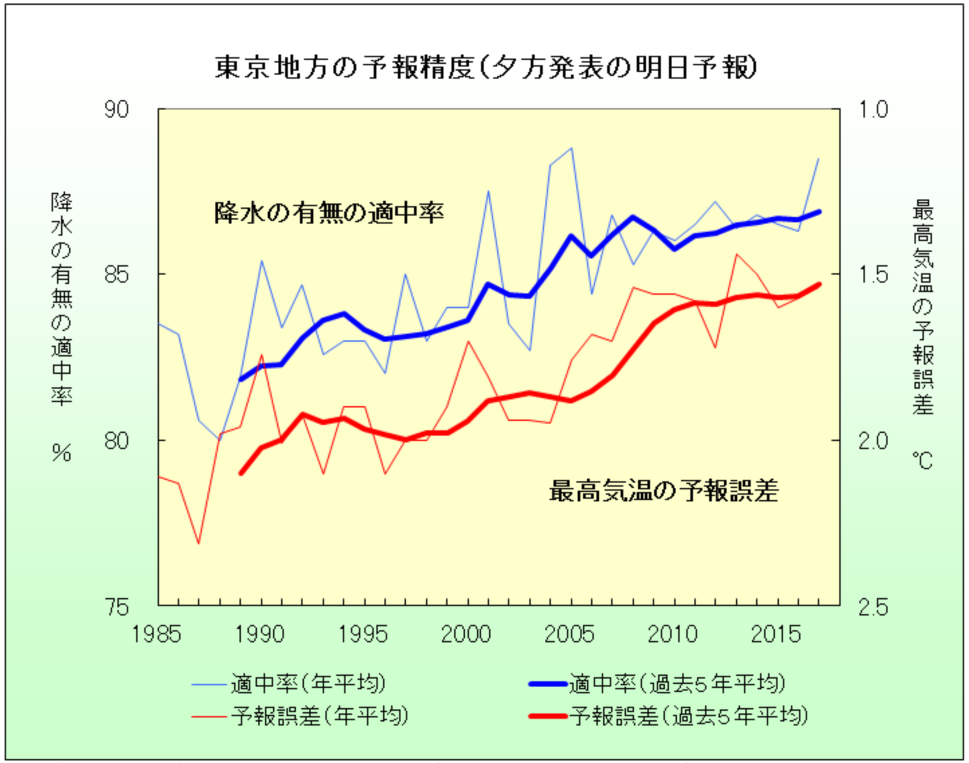
	cc1	cc2	cc3	cc4	cc5	cc6	cc7	cc8	cc9	cc10	cc11	cc12	cc13	cc14
1. add	IF	ID1	ID2	ID3	EX	MEM	WB							
2. add	IF	ID1	ID2	ID3	EX	MEM	WB							
3. bne		IF	ID1	ID2	ID3	EX	MEM	WB						
4. add							IF	ID1	ID2	ID3	EX	MEM	WB	
5. add							IF	ID1	ID2	ID3	EX	MEM	WB	
6. add								IF	ID1	ID2	ID3	EX	MEM	WB
7. add								IF	ID1	ID2	ID3	EX	MEM	WB

2-way superscalar adopting deeper pipeline executing instruction sequence with a branch

Branch predictor

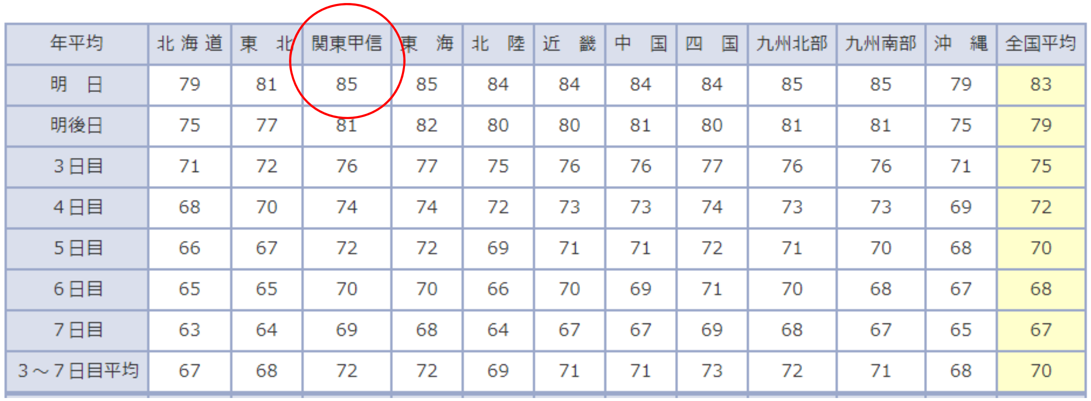
- A branch predictor is a digital circuit that tries to guess or predict which way (taken or untaken) a branch will go before this is known definitively.
  - A random predictor will achieve about a 50% hit rate because the prediction output is 1 or 0.
  - Let's guess the accuracy. What is the accuracy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

Prediction Accuracy of weather forecasts



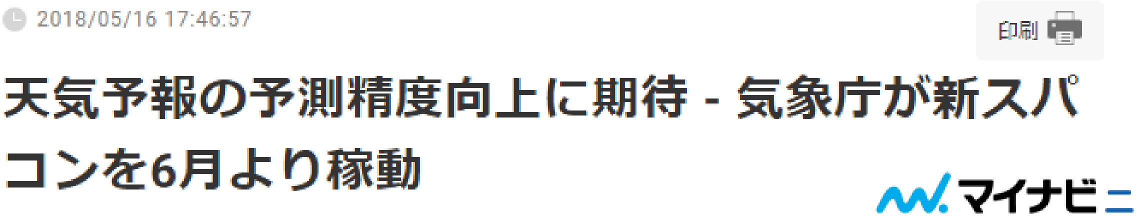
平成29年(2017年)までを表示しています。次の更新は平成31年(2019年)1月31日頃の予定です。

Tomorrow will be rainy?



年平均	北海道	東北	関東甲信	東海	北陸	近畿	中国	四国	九州北部	九州南部	沖縄	全国平均
明日	79	81	85	85	84	84	84	84	85	85	79	83
明後日	75	77	81	82	80	80	81	80	81	81	75	79
3日目	71	72	76	77	75	76	76	77	76	76	71	75
4日目	68	70	74	74	72	73	73	74	73	73	69	72
5日目	66	67	72	72	69	71	71	72	71	70	68	70
6日目	65	65	70	70	66	70	69	71	70	68	67	68
7日目	63	64	69	68	64	67	67	69	68	67	65	67
3～7日目平均	67	68	72	72	69	71	71	73	72	71	68	70

国土交通省
気象庁
Japan Meteorological Agency



2018/05/16 17:46:57

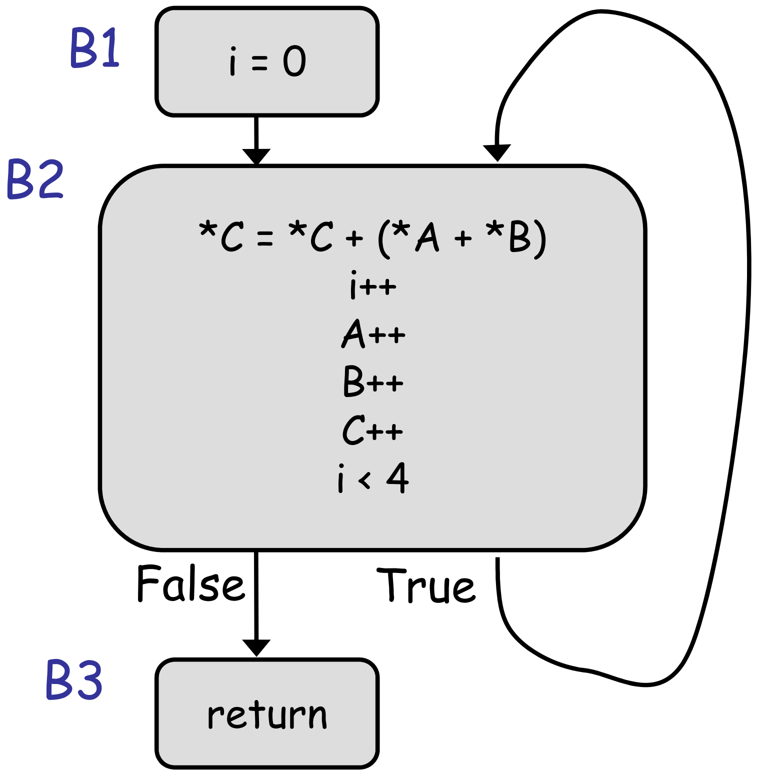
印刷

天気予報の予測精度向上に期待 - 気象庁が新スパコンを6月より稼動

マイナビニュース

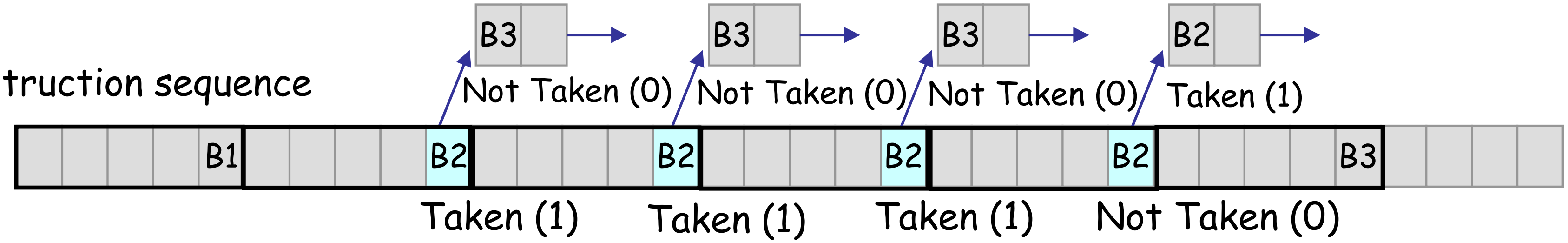
Sample program: vector add

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



Control flow graph

Executed instruction sequence

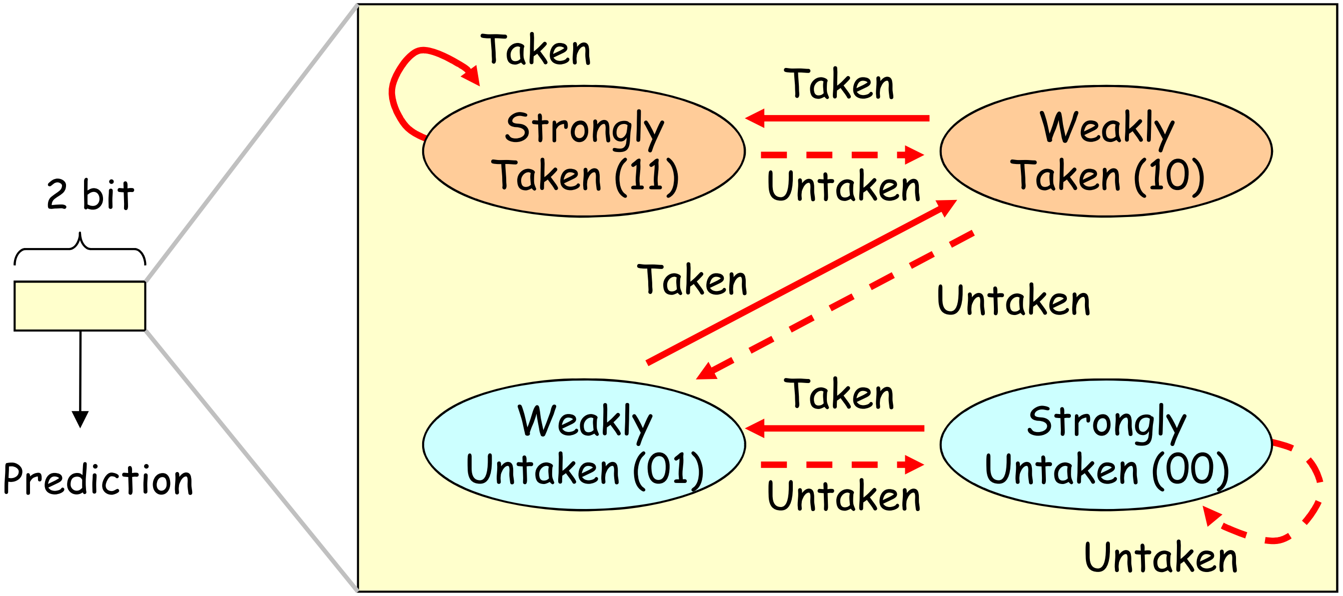


Simple branch predictor: Branch Always

- How to predict
  - It always predicts as 1.
- How to update
  - Nothing cause it does not use any memory.

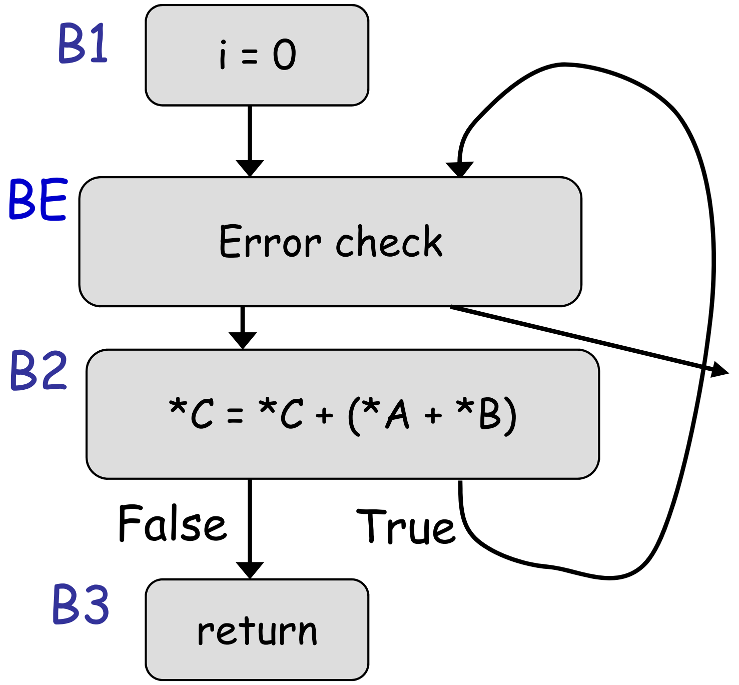
Simple branch predictor: 2bit counter

- It uses two bit register or a counter.
- Hot to predict
  - It predicts as 1 if the MSB of the register is one, otherwise predicts as 0.
- How to update the register
  - If the branch outcome is taken and the value is not 3, then increment the register.
  - If the branch outcome is untaken and the value is not 0, then decrement the register.



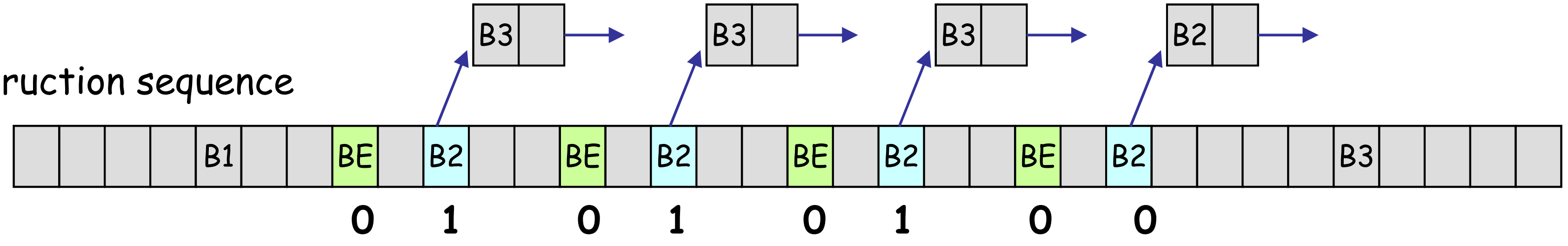
Sample program: vector add with two branches

```
#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]);
  }
}
```



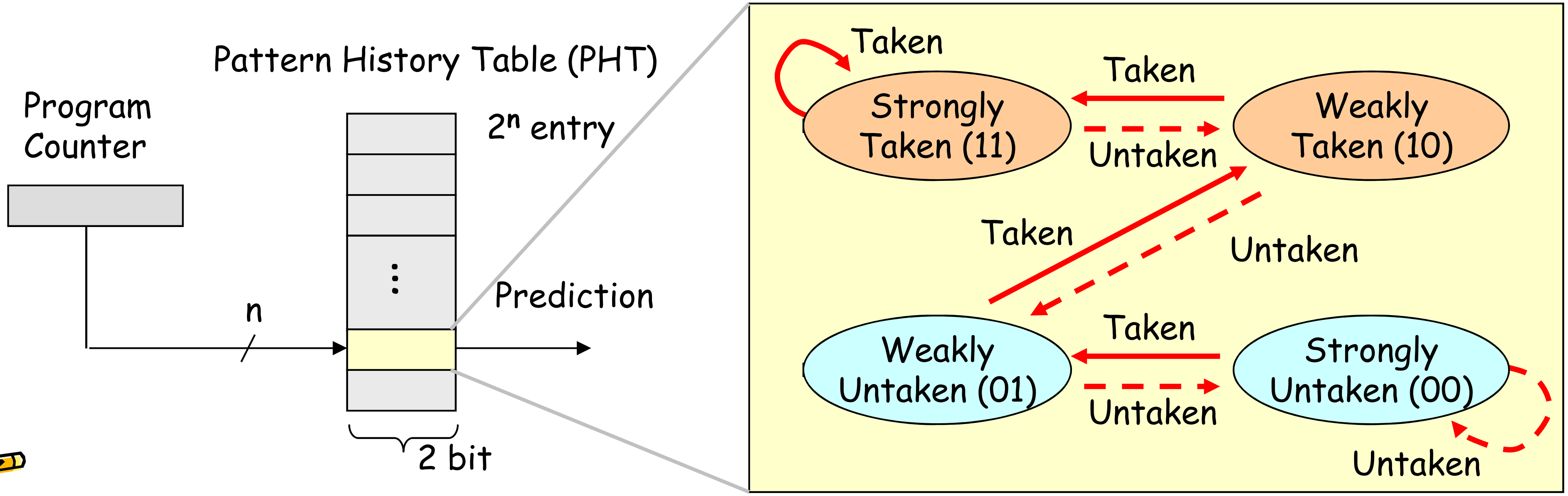
Control flow graph

Executed instruction sequence



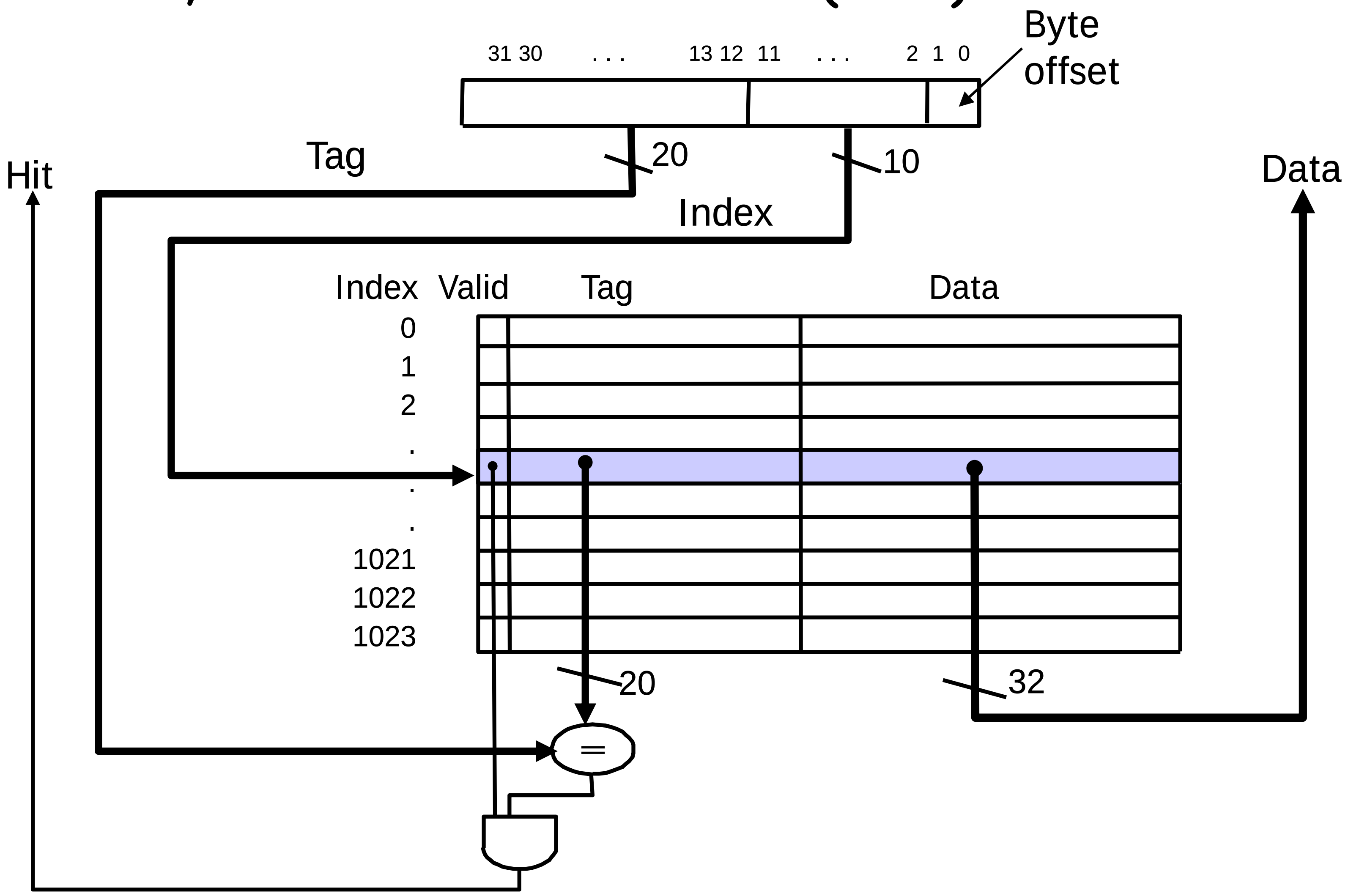
Simple branch predictor: bimodal

- 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 same manner as 2bit counter.



MIPS Direct Mapped Cache Example

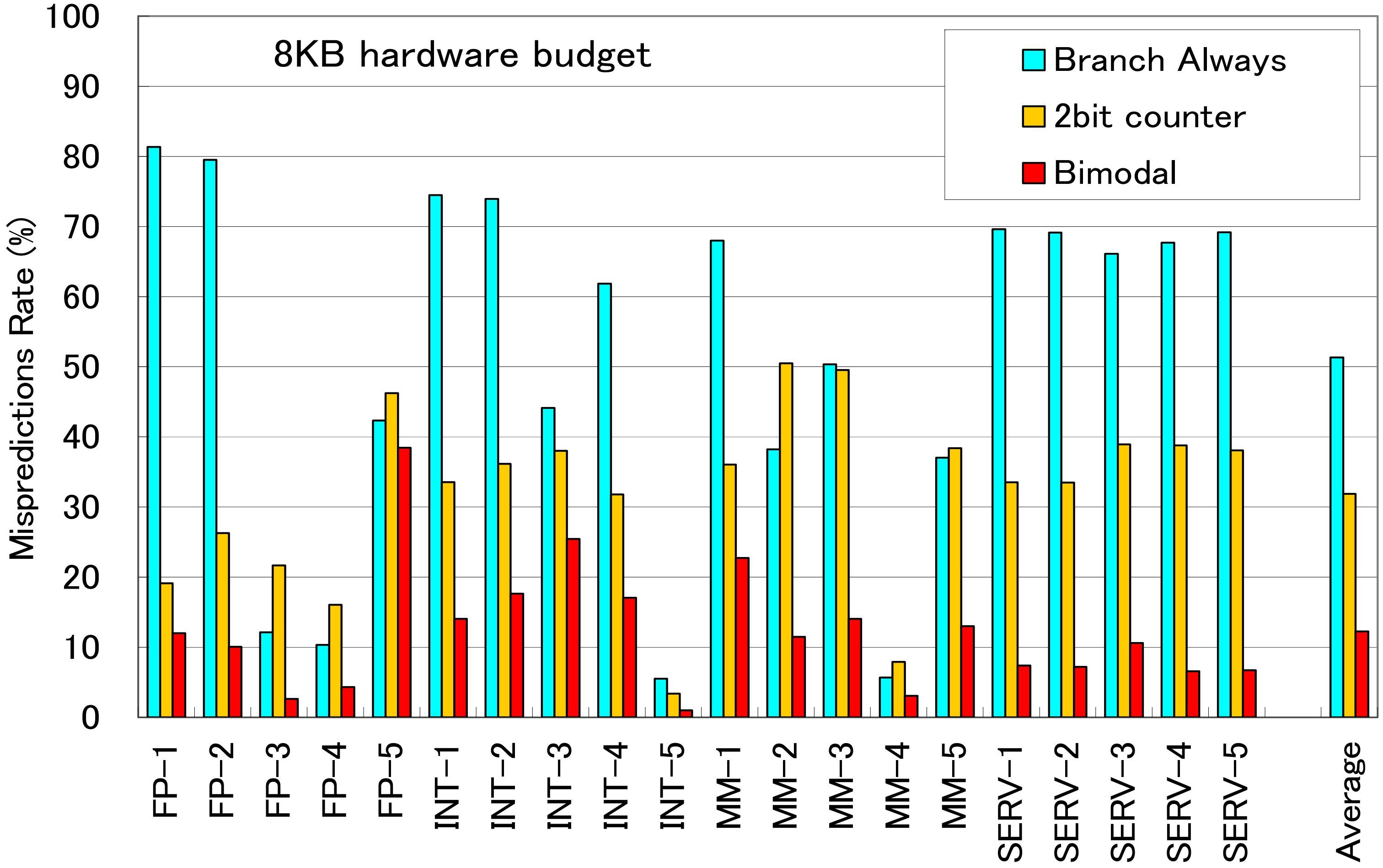
- One word/block, cache size = 1K words (4KB)



What kind of locality are we taking advantage of?

Prediction accuracy of simple branch predictors

- The accuracy of branch always is about 50%.
- The accuracy of bimodal predictor of 4KB memory is about 88%.



Benchmark for CBP(2004) by Intel MRL and IEEE TC uARCH.