《计算机系统结构》课程直播
2020. 5. 7 习题课

听不到声音请及时调试声音设备，可以下课后补签到

请将ZOOM名称改为“姓名”；

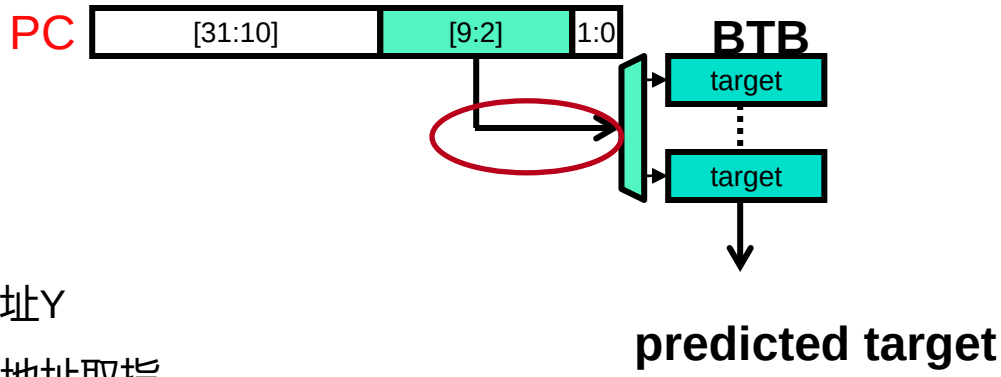
本节内容

- ILP概念复习
- 转移预测复习
- 多线程处理器复习

From : H&P Computer Architecture: A Quantitative Approach, Fifth Edition, （5th edition)

Branch Target Buffer: BTB

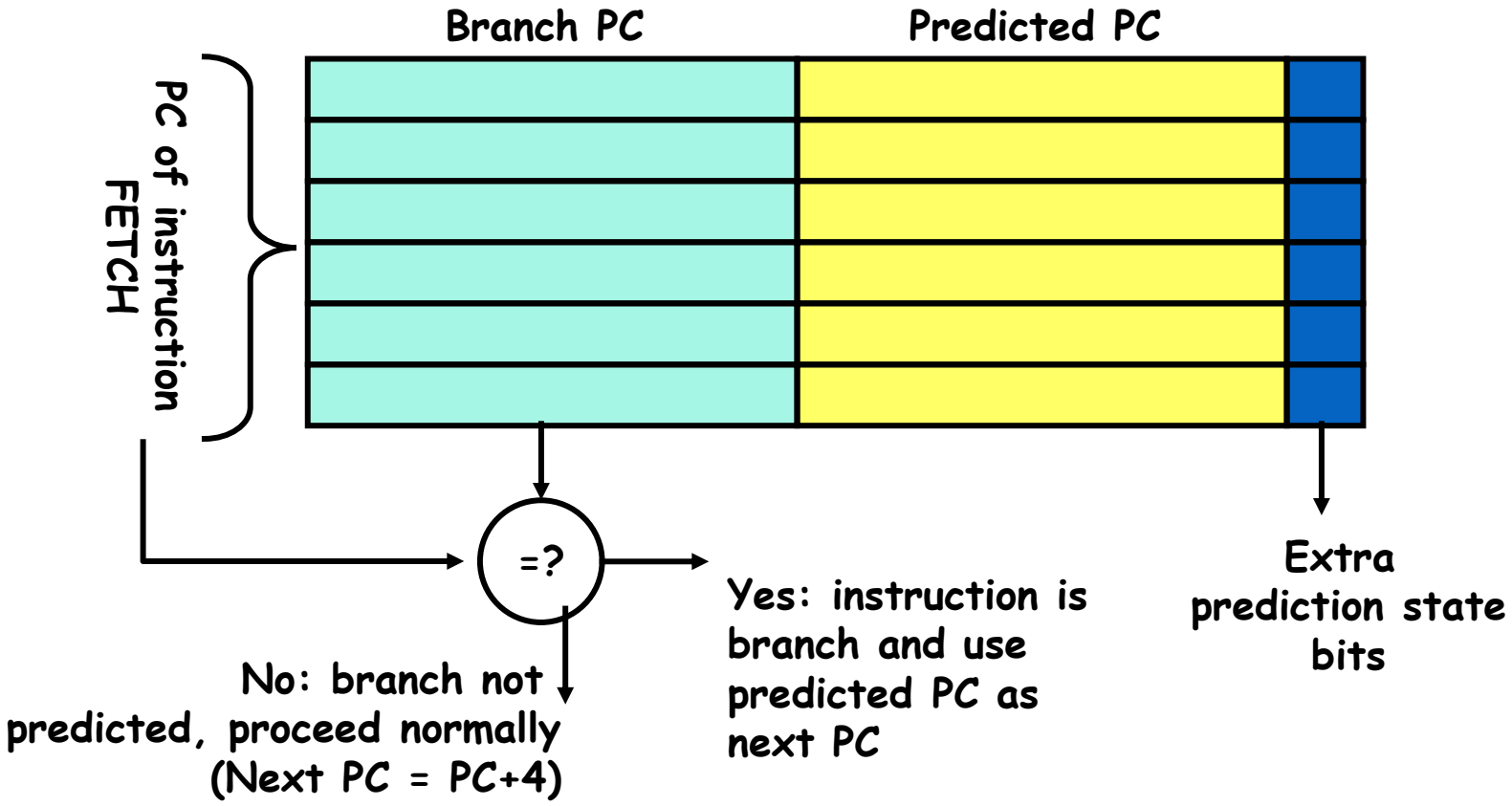
- **用历史预测未来**
  - 用硬件实现一张表
    转移历史表

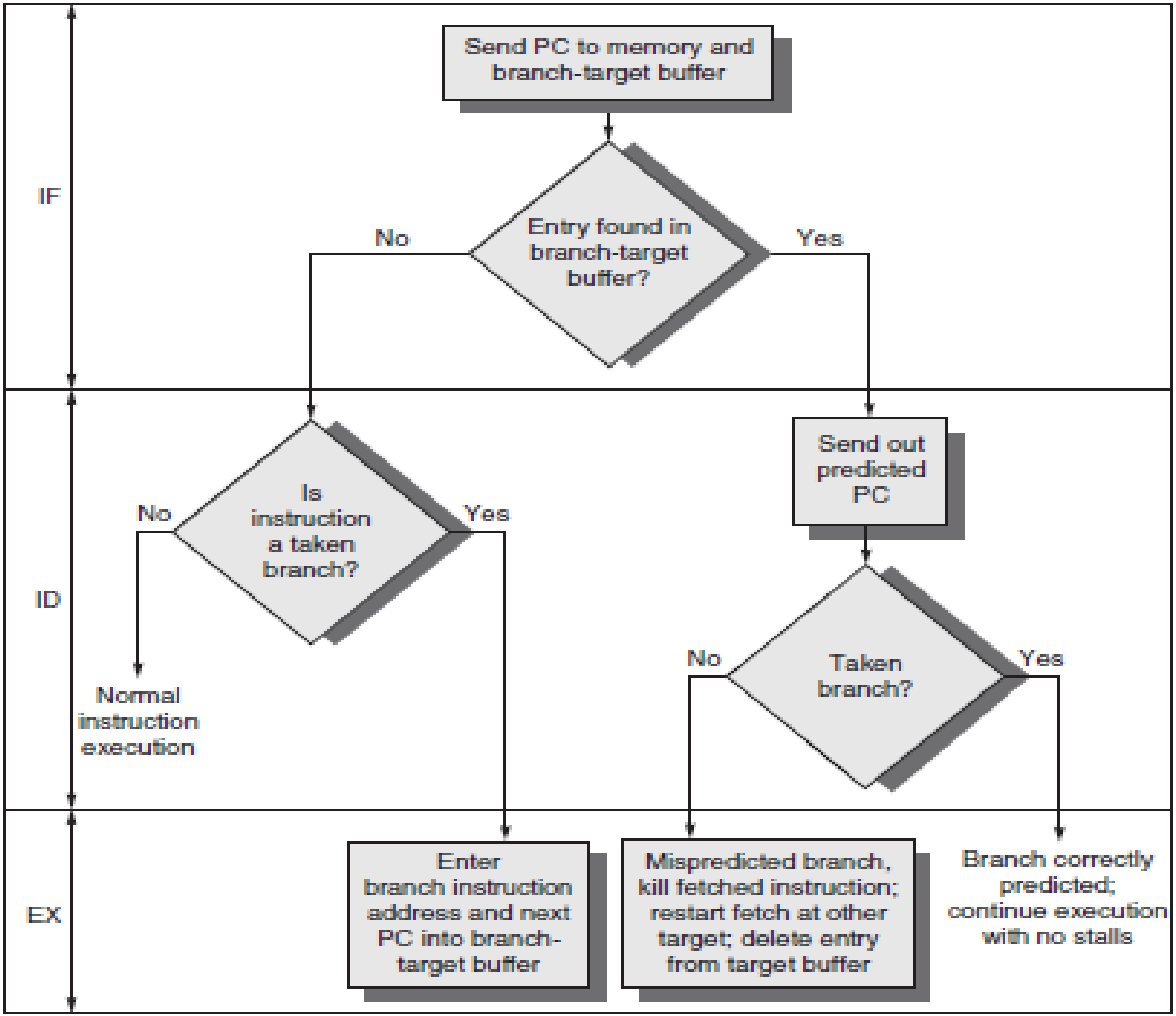


- **Branch target buffer (BTB)**:
  - 上次 branch X 转了、并转往地址Y
  - 如果在X地址取指，接下来在 Y地址取指
  - 用PC来检索BTB
  - 同名怎么办?
    - 两个PC最后几位相同
    - 没关系，只是预测

简单的动态预测：Branch Target Buffer (BTB)

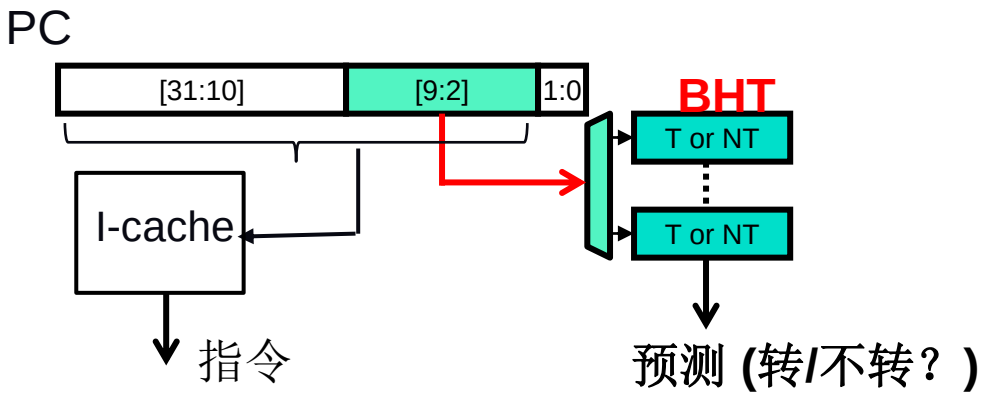
- 分支指令的地址作为BTB的索引，以得到分支预测地址
  - 必须检测分支指令的地址是否匹配，以免用错误的分支地址
  - 从表中得到预测地址
  - 分支方向确定后，更新预测的PC




IF
ID
EX
Send PC to memory and branch-target buffer
Entry found in branch-target buffer?
No
Yes
Is instruction a taken branch?
No
Yes
Normal instruction execution
Send out predicted PC
Taken branch?
No
Yes
Enter branch instruction address and next PC into branch-target buffer
Mispredicted branch, kill fetched instruction; restart fetch at other target; delete entry from target buffer
Branch correctly predicted; continue execution with no stalls

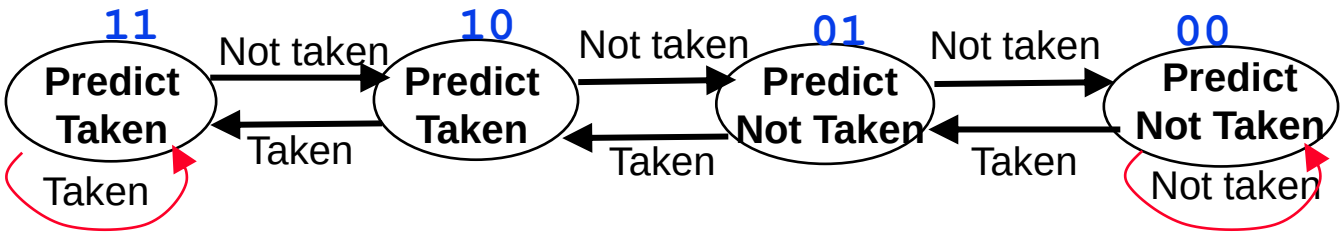
转移方向的预测：转 / 不转（Branch Outcome）？

- 学习过去，预测未来
- 90%的branch指令是有倾向性的
  - 例如for、while 语句
- 转移历史表
  - Branch History Table(BHT)



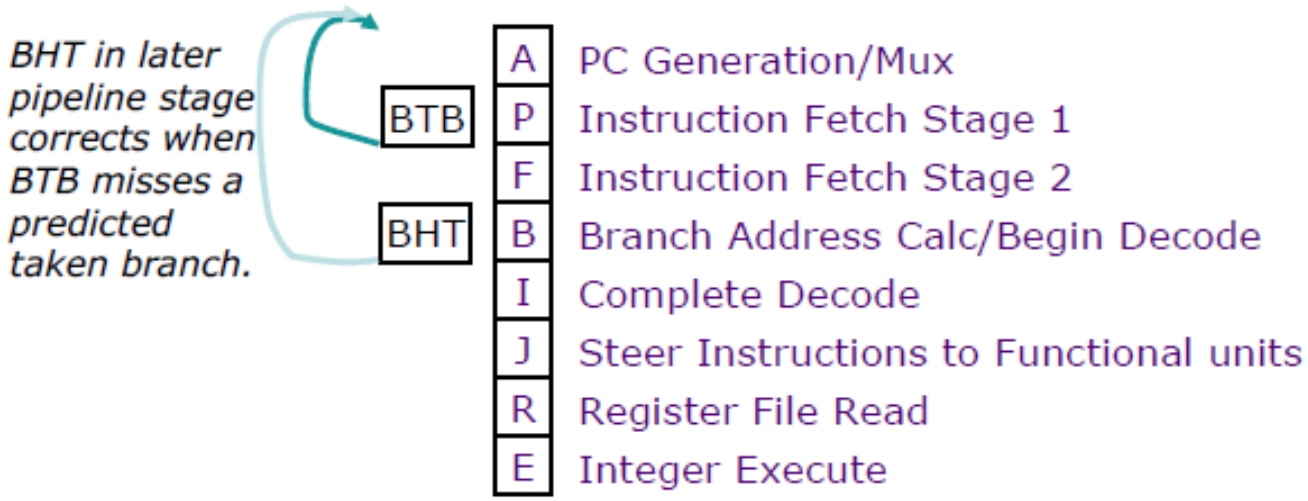
- 一边取指令一边查表预测
- PC索引位重合怎么办？
  - 没关系，这只是一个预测

两位预测器



00: strong NT
01: weak NT
10: weak T
11: strong T

课堂练习



	BTB Hit?	(BHT) Predicted Taken?	Actually Taken?	Pipeline bubbles
Conditional Branches	Y	Y	Y	
	Y	Y	N	
	Y	N	Y	Cannot occur
	Y	N	N	Cannot occur
	N	Y	Y	
	N	Y	N	
	N	N	Y	
	N	N	N	

课堂练习

- 我们在取指阶段增加一个转移目标缓冲器（branch target buffer: BTB):
- BTB表中,会为预测为“转移”的jump或branch 记录一对表项：entry_PC（当前指令的PC）, target_PC（转移目标指令的PC). 假设在BTB表中，转移目标即traget_PC一直是正确的（当然，转移方向的预测仍然有可能是错误的）；
- The BTB is looked up every cycle. If there is a match with the current PC, PC is redirected to the target_PC predicted by the BTB (unless PC is redirected by an older instruction); if not, it is set to PC+4.
- Fill out the following table of the number of pipeline bubbles (only for conditional branches). 填充以下表格，在各种情况下，流水线中浪费的周期数是多少？（仅仅考虑条件转移指令）

Correlating Branch Predicator

- 例如：

```
if (aa==2)
aa=0;
if (bb==2)
bb=0;
if (aa!=bb) {
```

➢ 翻译为MIPS

```
    SUBI R3,R1,#2
    BNEZ R3,L1          ; branch b1 (aa!=2)
    ADDI  R1,R0,R0      ;aa=0
L1: SUBI R3,R2,#2
    BNEZ R3,L2          ;branch b2(bb!=2)
    ADDI R2,R0,R0       ; bb=0
L2: SUBI R3,R1,R2       ;R3=aa-bb
    BEQZ R3,L3          ;branch b3 (aa==bb)
```

➢ 观察结果：

b3 与分支b2 和b1相关。

如果b1和b2都分支失败，则b3一定成功。

Sometimes 2-bit Saturating Counter Can't Work Well

- Local prediction

  For (i=1; i<=4; i++) {……}

  - Branch history pattern
    - 1110 1110 1110 1110 …

History pattern	prediction
111	0
110	1
101	1
011	1

Gshare predictor

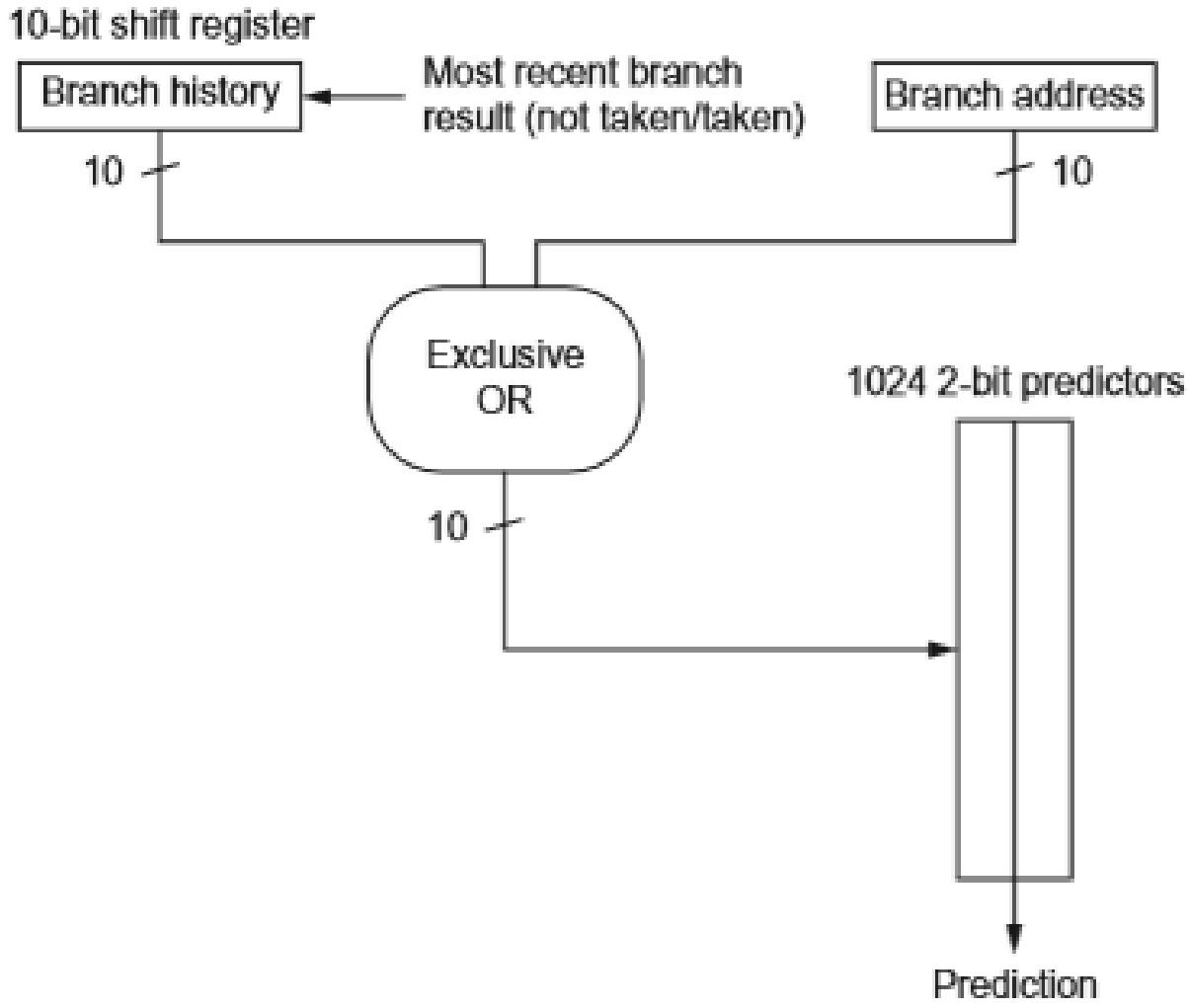
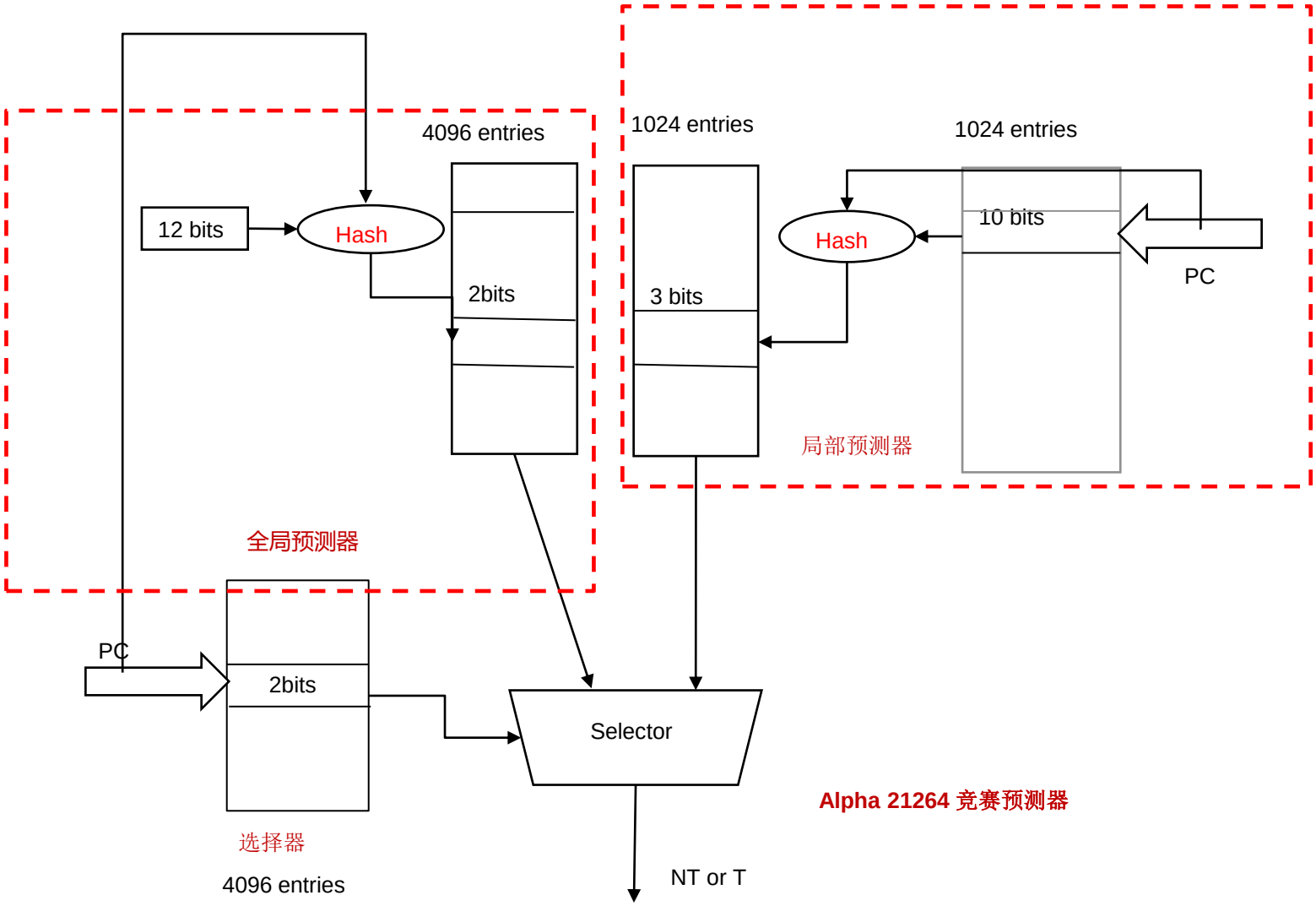


Figure 3.4 A gshare predictor with 1024 entries, each being a standard 2-bit predictor.

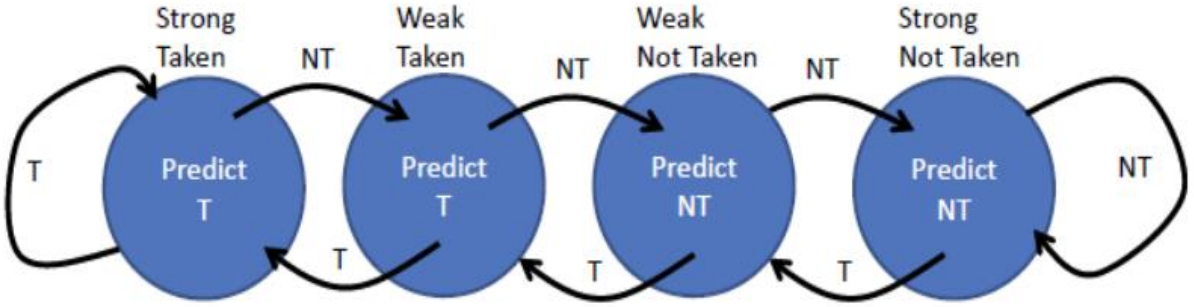
Alpha 21264 锦标赛预测器



Alpha 21264 竞赛预测器

习题

- 假设一个计算机系统中有一个1024 个表项（entry）的Branch History Table(转移历史表，BHT）。每个BHT表项是一个两位预测器，用于实现一个初始状态为SNT（strong not taken)的两位饱和计数器. 状态转换方式如下图：



- 在执行了以下代码之后，BHT中的状态是什么？每一条Branch指令的预测准确度（accuracy）是多少？
- NT: not taken; T: taken; X:未执行。

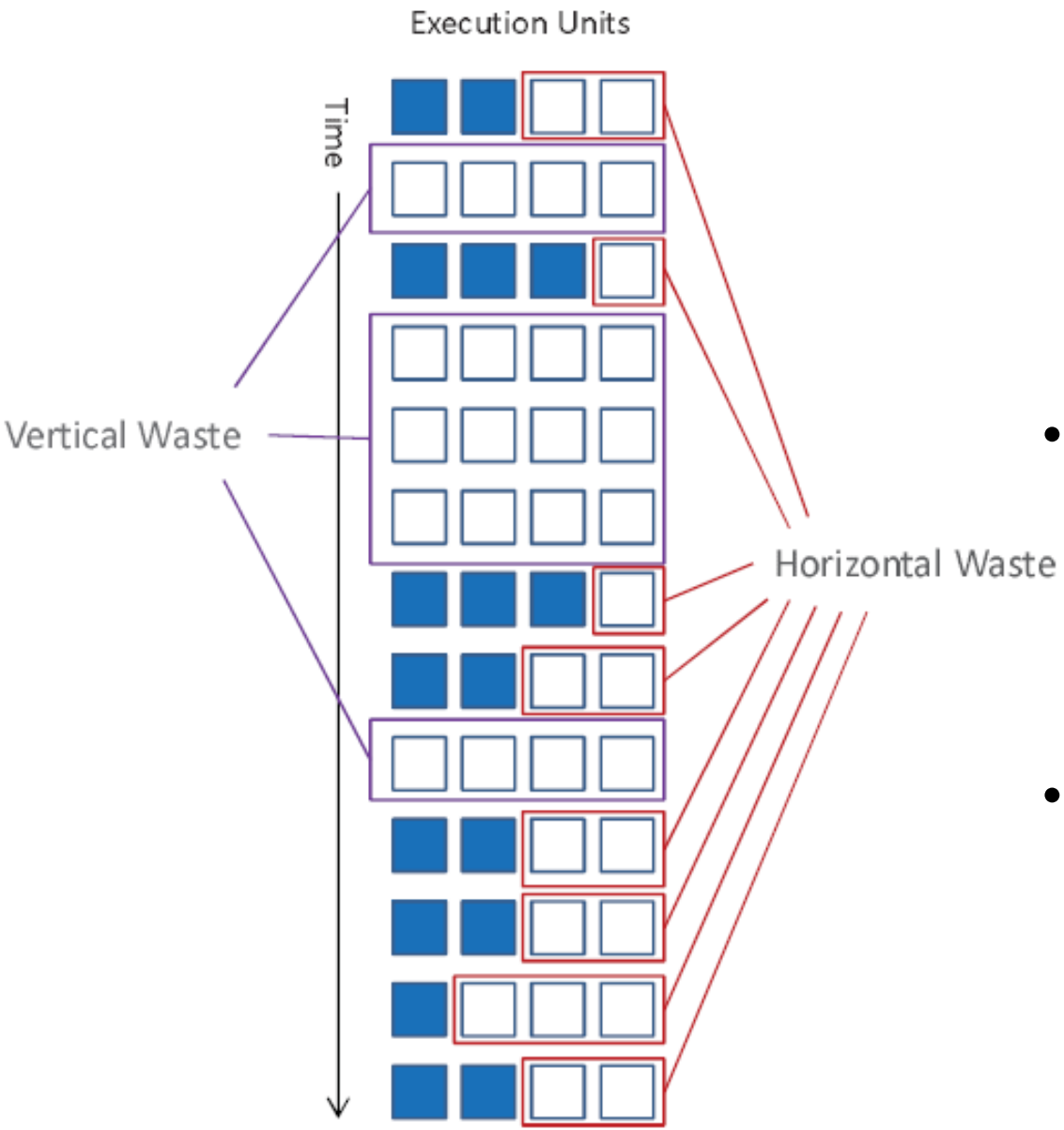
```
// Assume that at address 0x1000, there is an array of word sized integers
// with the following data [0x0, 0x0, 0x0, 0x0, 0x0, 0x1, 0x0, 0x1]

ADDI R7, R0, 7            //R0=0; ADDI: 加立即数
ADDI R11, R0, 0x1000
loop:
SUBI R7, R7, 1            // SUBI: 减立即数
ANDI R8, R7, 1            // ANDI: 和立即数相与
B_0:
BEQZ R8, l_1
LW R12, 0(R11)
B_1:
BEQZ R12, l_2
l_1:
ADD R9, R9, R16
l_2:
ADDI R11, R11, 4
B_3:
BNEZ R7, loop
```

解答：

- 下表中：
- initial代表预测器的初始状态，outcome代表对应branch指令实际的转移结果，after代表预测器转换后的新预测状态。

		Iteration 1		Iteration 2		Iteration 3		Iteration 4		Iteration 5		Iteration 6		Iteration 7		Accuracy
Name	Initial	Outcome	After	Outcome	After	Outcome	After	Outcome	After	Outcome	After	Outcome	After	Outcome	After	
b_0	SNT	T	NT	NT	SNT	T	NT	NT	SNT	T	NT	NT	SNT	T	NT	0.429
b_1	SNT	x	SNT	T	NT	x	NT	T	T	x	T	NT	NT	x	NT	0
b_3	SNT	T	NT	T	T	T	ST	T	ST	T	ST	T	ST	NT	T	0.571

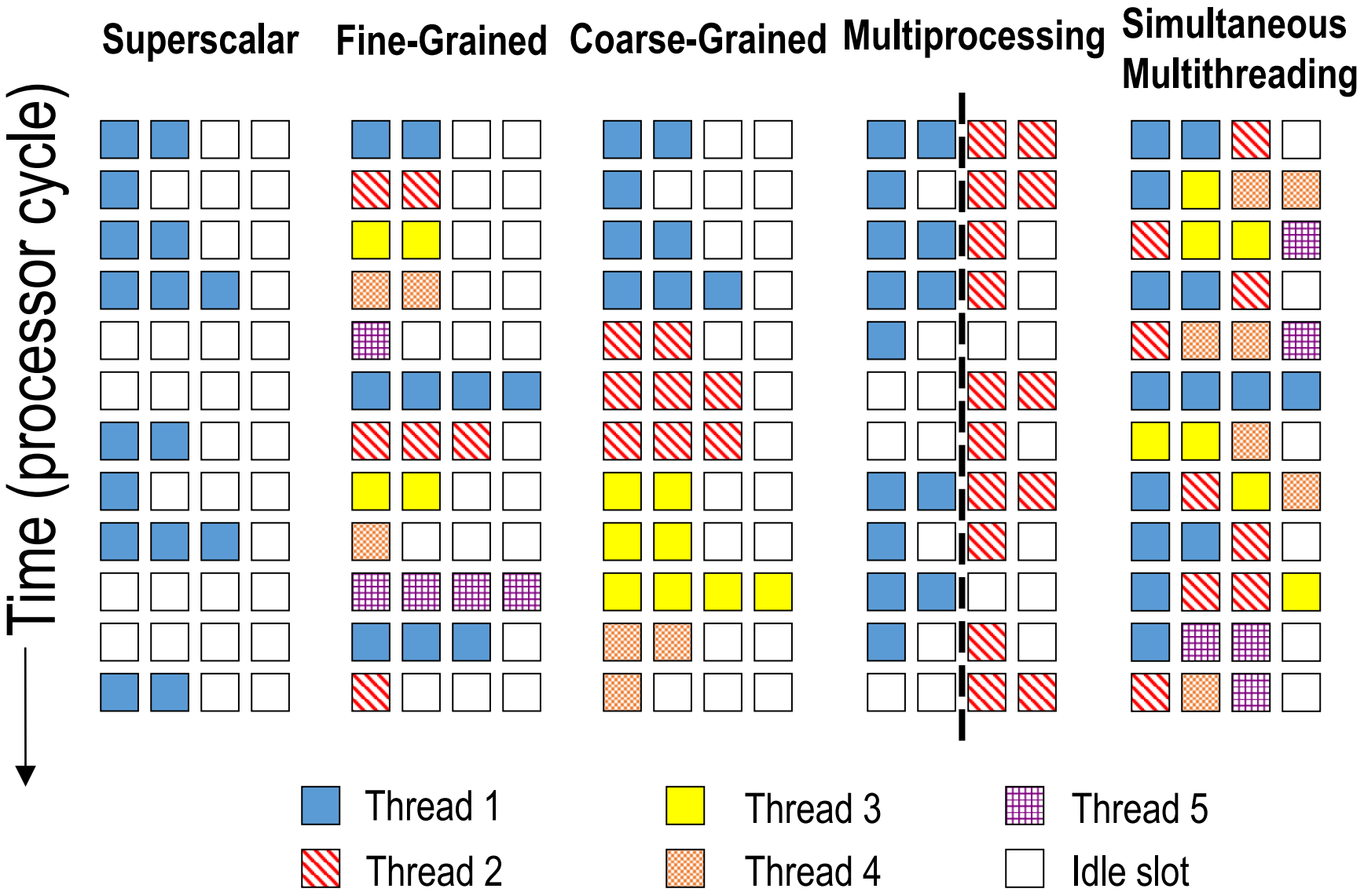
Superscalar Machine Efficiency



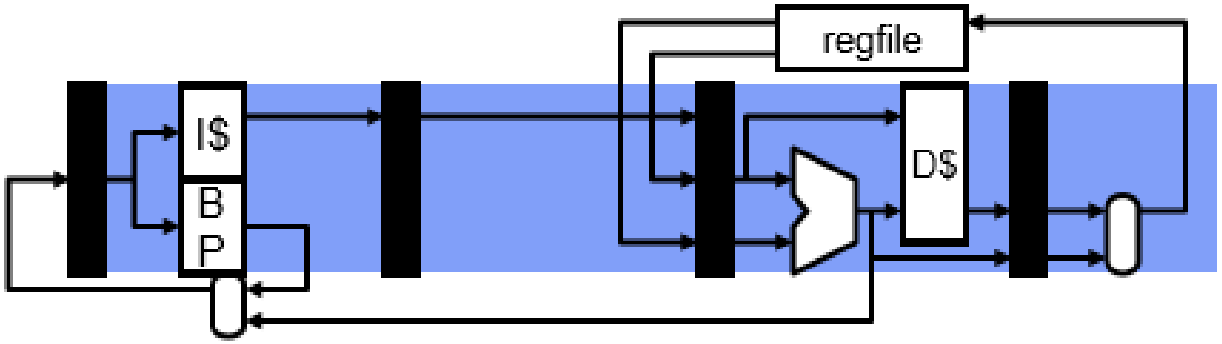
如何充分利用资源，提高并行度？
-- multithreading

- Multi-threading (MT) Improve utilization by multiplexing multiple threads on single core
- If one thread cannot fully utilize core? Maybe 2 or 4 can

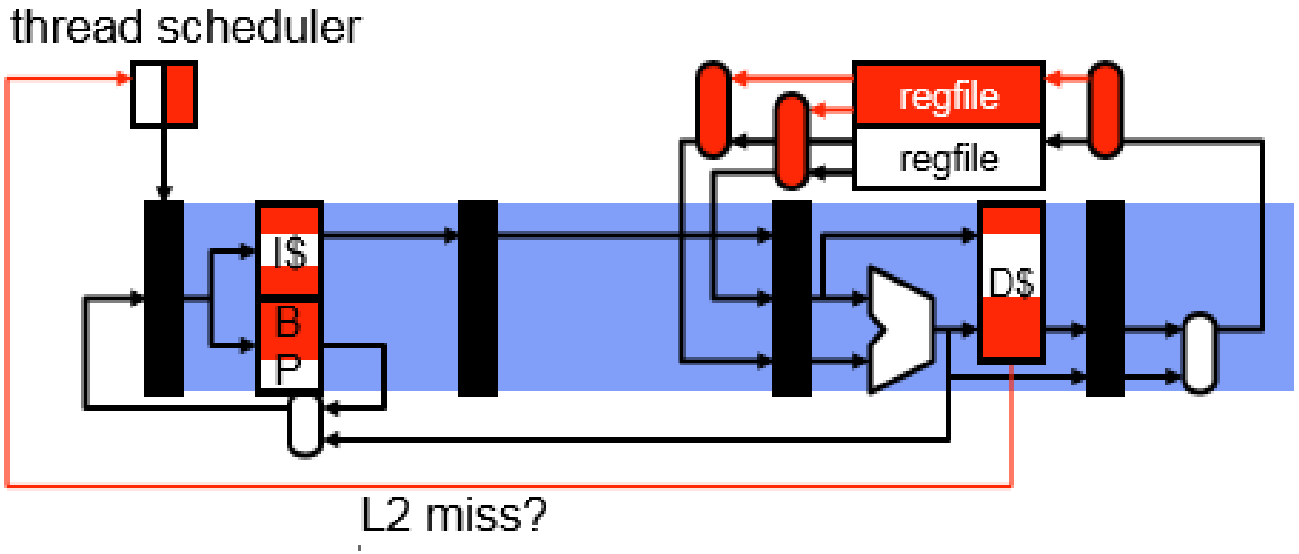
Summary: Multithreaded Categories



粗粒度多线程：Coarse-Grain Multithreading

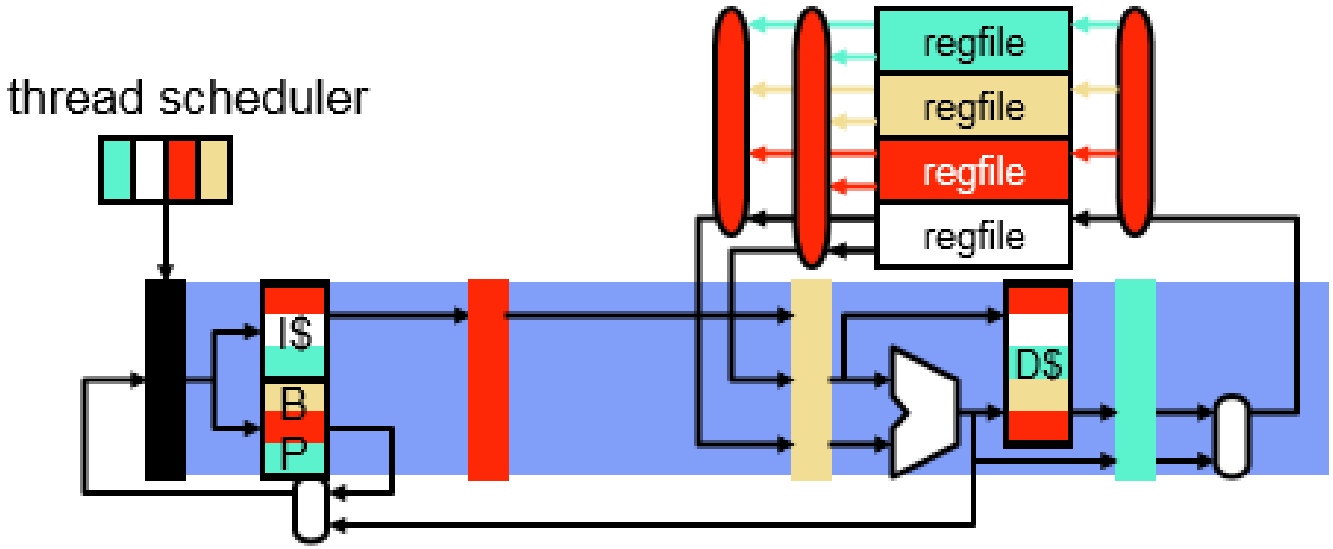


CGMT

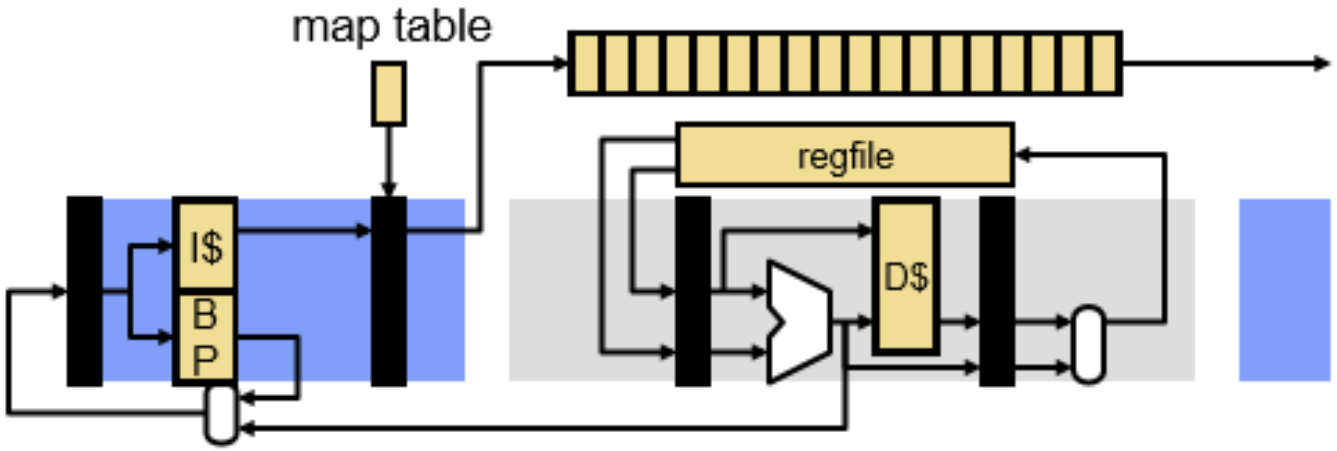


细粒度多线程：Fine-Grain Multithreading

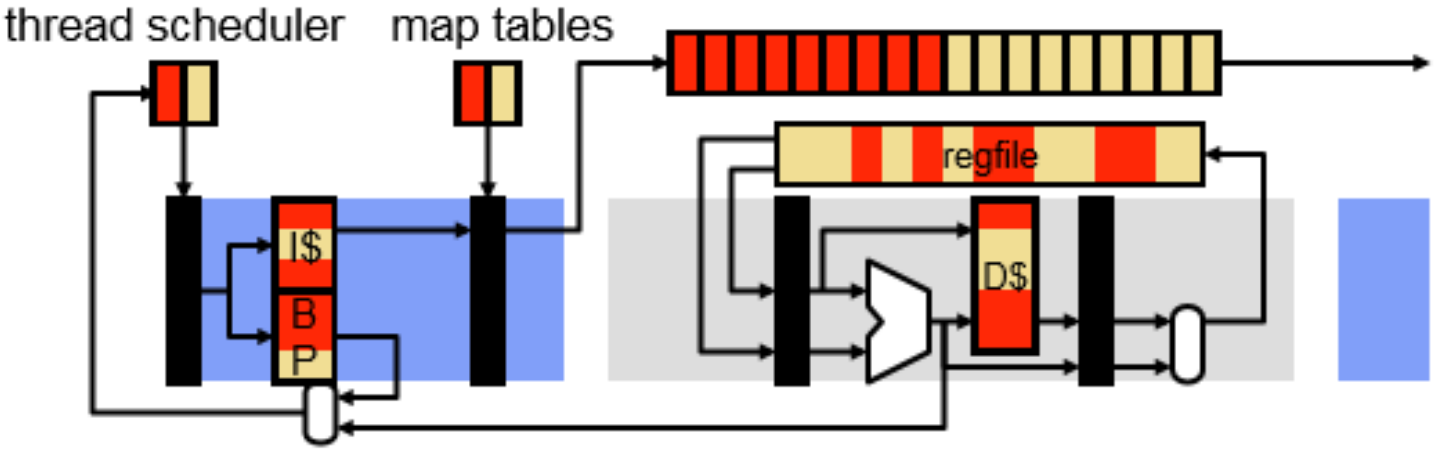
- FGMT
  - **Multiple threads in pipeline at once**
  - (Many) more threads



同时多线程：Simultaneous Multithreading



- SMT
  - Replicate map table, share (larger) physical register file



下一节

- 数据级并行性
- Data-Level Parallelism in Vector, SIMD, and GPU Architectures

From : H&P Computer Architecture: A Quantitative Approach, Fifth Edition, （5th edition)

谢　谢！