## Scoreboard Method

Functional Unit Status Table [Add $R_2$,$R_3$,$R_4$]

| Unit ID | Unit Name | Busy | Destination Register $R_d$ | Source Registers | | | |
|---|---|---|---|---|---|---|---|
| | | | | $R_{S1}$ | Ready | $R_{S2}$ | Ready |
| 1 | Load/Store | NO | | | | | |
| 2 | Multiplier | YES | R5 | R1 | YES | R2 | YES |
| 3 | Adder_1 | NO | | | | | |
| 4 | Adder_2 | YES | R2 | R2 | YES | R5 | NO |



## Scoreboard Method

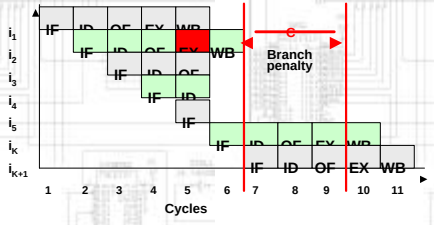
Destination Register Status Table [Add $R_2$,$R_3$,$R_4$]

| | R1 | R2 | R3 | R4 | R5 | R6 |
|---|---|---|---|---|---|---|
| Unit Id | | 4 | | | 2 | |



## Control Hazards

- All branch instructions have the potential to drain the pipe, imposing a branch penalty c which is proportional to the overall pipelength






## Branching

- Initially, of course , we cannot anticipate branches
- Once encountered we can preserve useful information:
  - Branch location in memory (PC)
  - Branch history (ie. taken or not)
  - Branch category

## Branch Category

- Unconditional (JUMP, GOSUB)
- Loop branch
  - If we branch back not for from the branch location we probably have a loop body
  - Preserve it in the IF stage
- Conditional branch
  - Use history to decide between target or no branch



## Branch Penalty

Average number of cycles per inst.

with $p_b$ = probability instruction is a branch

$t_{ave} = p_b$(average number of cycles per branch instruction) + $(1- p_b)$(average number of cycles per non-branch inst.)

Average number of cycles per branch instruction

If target path = 1+ c cycles with c as branch penalty

If sequential = 1 cycles

With $p_t$ = probability branch is taken

Average number of cycles per branch instruction = $p_t(1+c) + (1- p_t)1$

$t_{ave} = p_b(p_t(1+c) + (1- p_t)1) + (1- p_b)(1)$

$= p_b p_t c + 1$



## Branch Penalty

$t_{ave} = p_b p_t c + 1$

Typically $0.1 < p_b < 0.3$ and $0.6 < p_t < 0.7$

For a 5-stage pipe with c=3

$t_{ave} = p_b p_t c + 1 = 0.2 * 0.65 * 3 + 1 = 1.39$

Efficiennncy = $1 / t_{ave} * 100 = 72\%$



## Impact of Branches

- Techniques for reducing the impact of branches
  - Branch Prediction
  - Multiple Prefetching
  - Delayed Branching

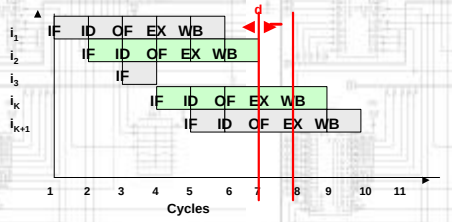
## Branch Prediction

- Predict the outcome of the branch before it is executed
- Incorrect predictions may increase penalties
- Static prediction –
  - for example, always assume a branch will be taken
- Dynamic –
  - use branch target buffer (BTB) (see 3BA4)



## Control Hazards

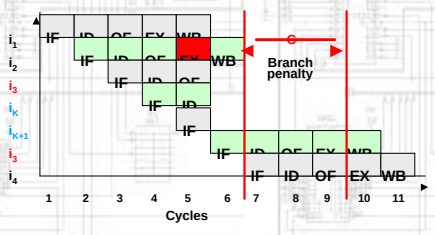
Penalty for a correctly chosen target path






## Control Hazards

Penalty for a incorrectly chosen target path






## Performance effect of Branch Prediction

with $p_t$ = probability branch is taken

Average number of cycles per branch instruction =

$p_t(1+c) + (1- p_t)1$

with $p_r$ = probability of right prediction ->

Average number of cycles per branch instruction =

$$p_r[\underbrace{p_t(1+d)}_{\text{Target}} + \underbrace{(1- p_t)1}_{\text{Sequential}}] + (1- p_r)[\underbrace{p_t(1+c)}_{\text{Target}} + \underbrace{(1- p_t)(1+c)}_{\text{Sequential}}]$$

Correct Prediction — Incorrect Prediction