

Control Details for Tomasulo Algorithm

- Following table details the controls necessary to enforce the Tomasulo algorithm.

Instruction State		Wait Condition	Control
Issue	FP Operation	Empty 'r' Station	IF (RegisterStat[rs].Qi≠0) {RS[r].Qj ← RegisterStat[rs].Qi } ELSE {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0}; IF (RegisterStat[rt].Qi≠0) { RS[r].Qk ← RegisterStat[rt].Qi } ELSE {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0}; RS[r].Busy ← yes; RegisterStat[rd].Qi = r;
	Load/ Store	Empty 'r' Buffer	IF (RegisterStat[rs].Qi≠0) {RS[r].Qj ← RegisterStat[rs].Qi } ELSE {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0}; RS[r].A ← imm; RS[r].Busy ← yes;
	Load Only		RegisterStat[rt].Qi = r;
	Store Only		IF (RegisterStat[rt].Qi≠0) {RS[r].Qk ← RegisterStat[rs].Qi } ELSE {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0};
Execute	FP Operation	(RS[r]Qj = 0) & (RS[r]Qk = 0)	Compute result: operands are in Vj and Vk
	Load-Store (1)	RS[r].Qj = 0 & r is head of load- store queue	RS[r].A ← RS[r].Vj + RS[r].A;
	Load-Store (2)	Load-Store (1) complete	Read from MEM[RS[r].A]
Write Result	FP Operation or Load	Execution complete at 'r' & CDB available	∀x(IF (RegisterStat[rt].Qi = r) THEN {Regs[x] ← result; RegisterStat[x].Qi ← 0}) ∀x(IF (RS[x].Qj = r)

		$\text{THEN } \{RS[x].V_j \leftarrow \text{result}; RS[x].Q_j \leftarrow 0\}$ $\forall x(\text{IF } (RS[x].Q_k = r)$ $\text{THEN } \{RS[x].V_k \leftarrow \text{result}; RS[x].Q_k \leftarrow 0\})$ $RS[r].\text{Busy} \leftarrow \text{no};$
Store	Execution complete at 'r' & $RS[r].Q_k = 0$	$\text{MEM } [RS[r].A] \leftarrow RS[r].V_k;$ $RS[r].\text{Busy} \leftarrow \text{no};$

- Notes,
 - 'rd' denotes destination, 'rs' and 'rt' source register ID.
 - 'imm' is the sign extended immediate address;
 - 'r' is the reservation station / buffer;
 - 'RS' is the reservation station data structure;
 - 'result' is the value returned by an FP or load unit;
 - 'RegsiterStat' is the register status data structure;
 - 'Regs' is the register file;

Summary

- In order to avoid WAR and RAW hazards the Tomasulo algorithm combines,
 - Register renaming with,
 - Source operator buffering.
- Advantages
 - Minimizes sensitivity to unscheduled code;
 - Fundamental requirement when multi-issue designs are sought (see later lecture);
 - Supports speculative execution (see later lecture).
- Disadvantages
 - Sensitive to branch prediction (i.e. branch prediction is a necessity);
 - High hardware overhead;
 - CDB represents a bottleneck.