Credits: Dr. Yousif (Intel), Kubi@UCB, Asanovic @ MITs

Addendum to Handout 12

Shoukat Ali

shoukat@umr.edu

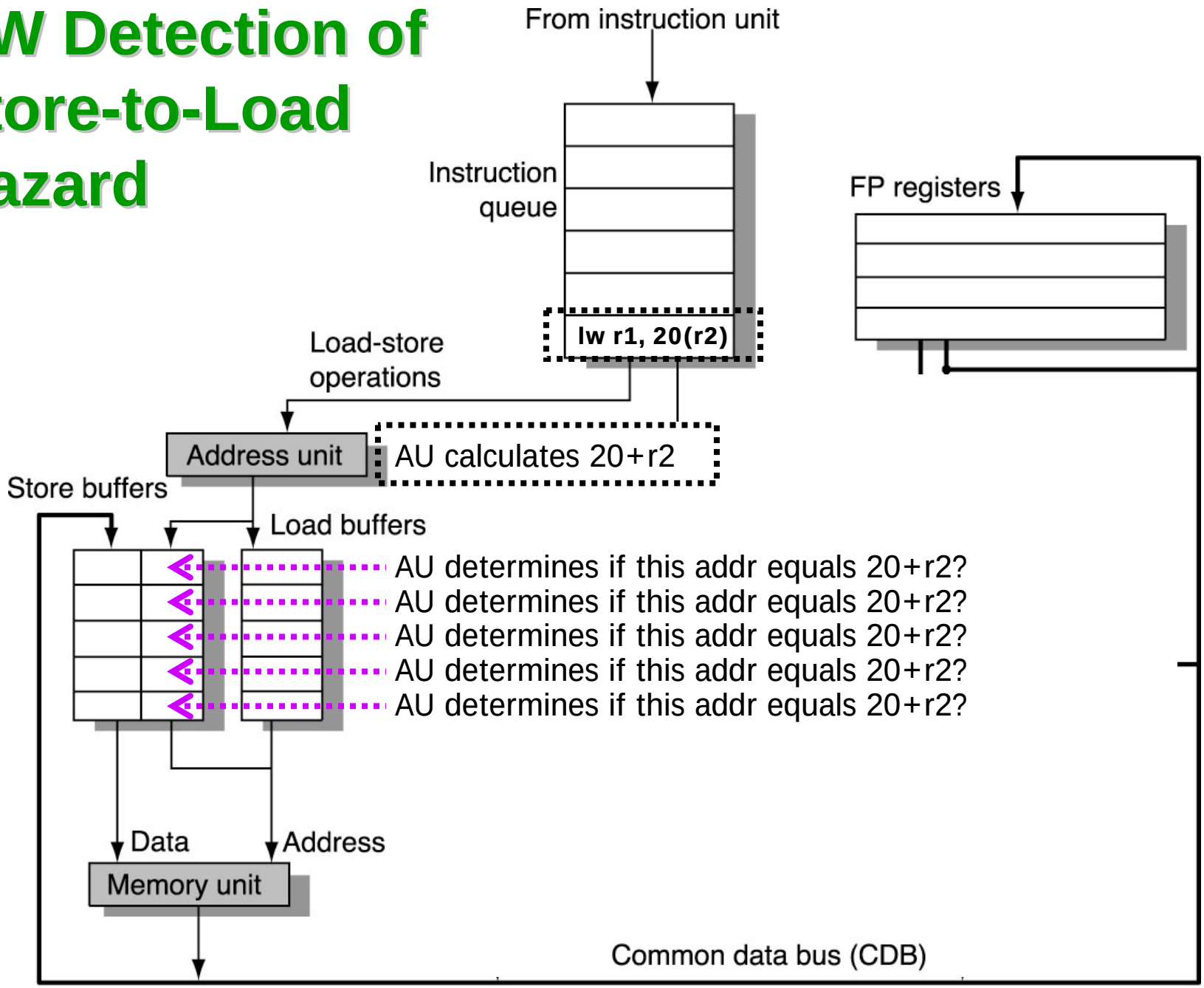
UMR
UNIVERSITY OF MISSOURI-ROLLA
The Name. The Degree. The Difference.

Data Hazards That A Compiler Cannot See

- okay to re-order if (16+R2) is not equal to (20+R4)
- otherwise this relative ordering must never be changed for correct program execution
  - a potential RAW hazard exists

- how does Tomasulo's algorithm ensure that "relative ordering" of a store-load sequence is not changed?

HW Detection of Store-to-Load Hazard



HW Detection of Store-to-Load Hazard

- before sending a load instruction to its reservation station (called a load buffer), do this:
  1. calculate the memory address of the load
  2. compare it with the memory address of all store instructions currently in the store buffers (reservation stations for store instructions)
  3. if there is an address match, do not send the load instruction to load buffer
- above three steps performed by the address unit
- essentially, a load instruction waits in the instruction queue until all stores issued before have completed

Instruction state	Wait until	Action or bookkeeping
Issue FP Operation	Station r empty	`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 or Store	Buffer r empty	`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) and (RS[r].Qk = 0)	Compute result: operands are in Vj and Vk
Load-Store step 1	RS[r].Qj = 0 & r is head of load-store queue	`RS[r].A ← RS[r].Vj + RS[r].A;`
Load step 2	Load step 1 complete	Read from Mem[RS[r].A]
Write result FP Operation or Load	Execution complete at r & CDB available	`∀x(if (RegisterStat[x].Qi=r) {Regs[x] ← result; RegisterStat[x].Qi ← 0}); ∀x(if (RS[x].Qj=r) {RS[x].Vj ← result;RS[x].Qj ← 0}); ∀x(if (RS[x].Qk=r) {RS[x].Vk ← result;RS[x].Qk ← 0}); RS[r].Busy ← no;`
Store	Execution complete at r & RS[r].Qk = 0	`Mem[RS[r].A] ← RS[r].Vk; RS[r].Busy ← no;`

Figure 3.5 Steps in the algorithm and what is required for each step. For the issuing instruction, rd is the destina-