Lecture 10 slides

CE/CZ 3001:
Advanced Computer Architecture

(Module 4: Instruction Level Parallelism(ILP))

Dr Smitha K. G.
School of Computer Science
And Engineering

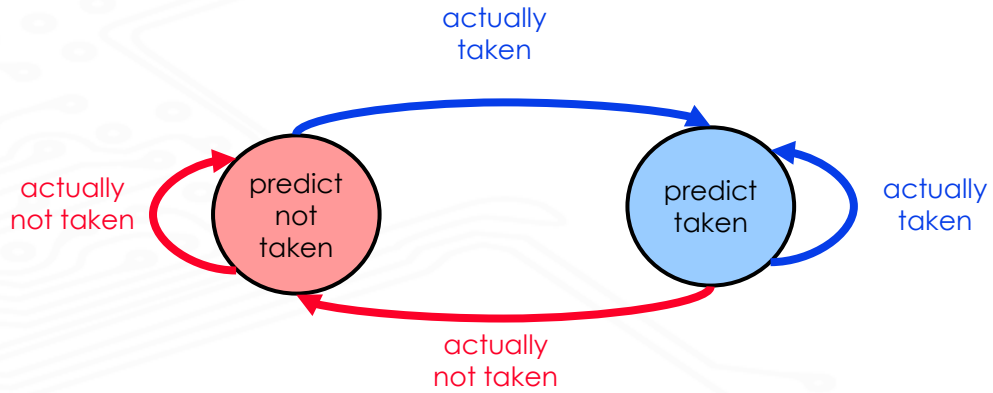
Summary of Pre video

- Dynamic branching and Basic of multi-issue processing.
- Limitation o f scalar pipeline and Motivation for superscalar processors.
- Concept of superscalar processing.
- Basics of superscalar pipeline.
- Instruction execution in a superscalar processor.
- Basics of VLIW processors.

Dynamic Prediction- Extra hardware required

Scheme 1: Single T bit (Last time predictor)

- T is set to 1 when a branch is confirmed taken , 0 when not.

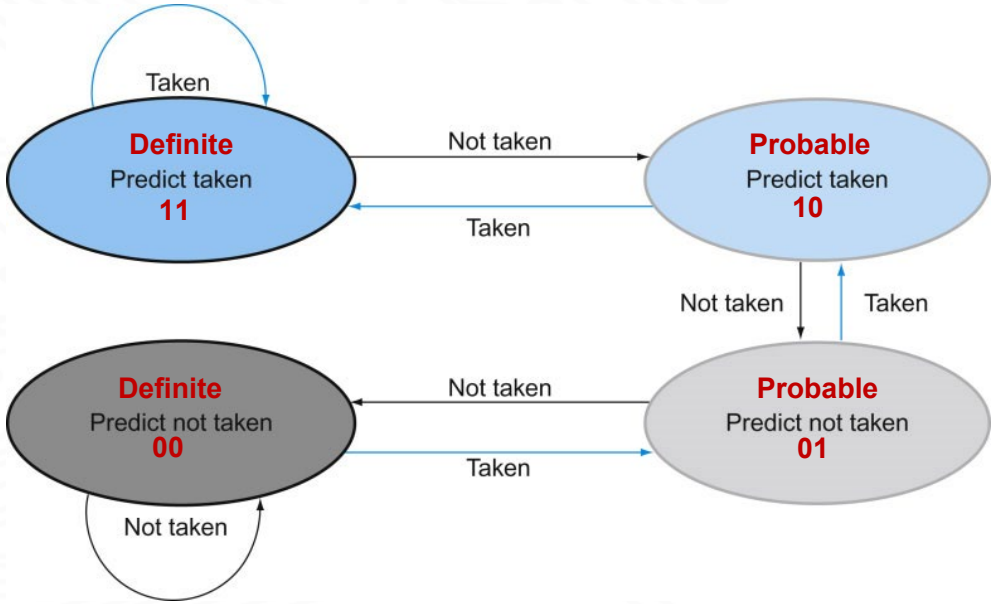


Example:

Assume single bit predictor for branch b1. Initial value of predictor to be 0. Then T_{b1} predicts that the branch is not taken.

branch b1 taken, T_{b1} =1 (predict next branch taken)
branch b1 not taken, T_{b1} =0 (predict next branch not taken)

Dynamic Prediction (two bit prediction)

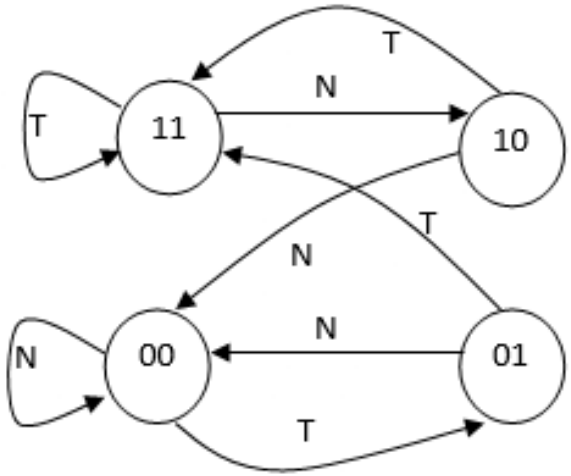


Scheme 2: 2 bit prediction uses the result of last two branches to predict instead of just the last branch.

- TNTNTNTNTNTNTNTNTNTN → 50% accuracy
- (assuming initial to probable (weakly taken)
- Disadvantage: More hardware

Example from 2017/2018 sem1 paper

3. (a) Consider the following repeating sequence of actual outcome for a branch (N N T N T N). Here 'N' means that the branch is not taken and 'T' means that branch is taken. Assume that there is only one branch instruction in the program. The predictors are initialized to 'weakly taken' stage.



11 → Strongly taken

10 → Weakly taken

01 → Weakly not taken

00 → Strongly not taken

Figure Q3a

i. Find the prediction accuracy of two-bit predictor shown in Figure Q3a by properly indicating the prediction decision at each stage.

(7 marks)

ii. Compare the accuracy with an always not taken static predictor and comment on the best choice of prediction for the above case.

(4 marks)

state	10											
prediction												
Actual												

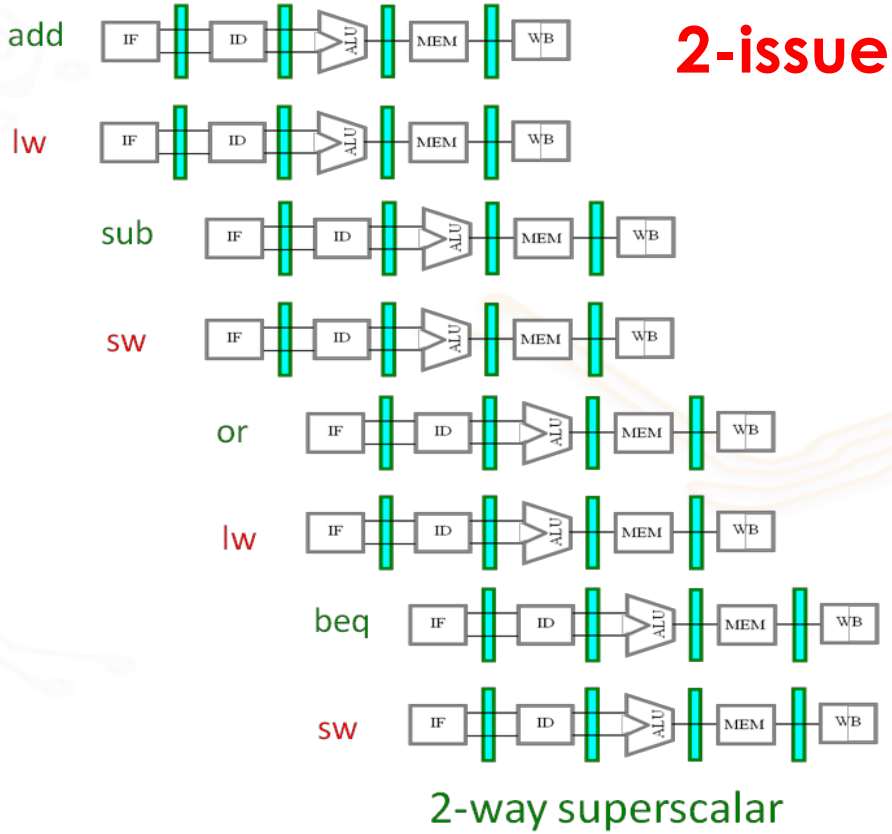
Dynamic Prediction (Part 2/2)

- **Scheme 3: Bimodal prediction** uses a counter and the state of that counter determines the prediction.
- Generalized scheme of 3 bit predictor.
- The counter is incremented if branch is taken.
- The counter is decremented if branch not taken.
- Counters saturate (no wraparound). The speculation decision is based on the most significant bit: if MSB is 1, then counter is above half way.

000 → 001 → 010 → 011 → 100 → 101 → 110 → 111

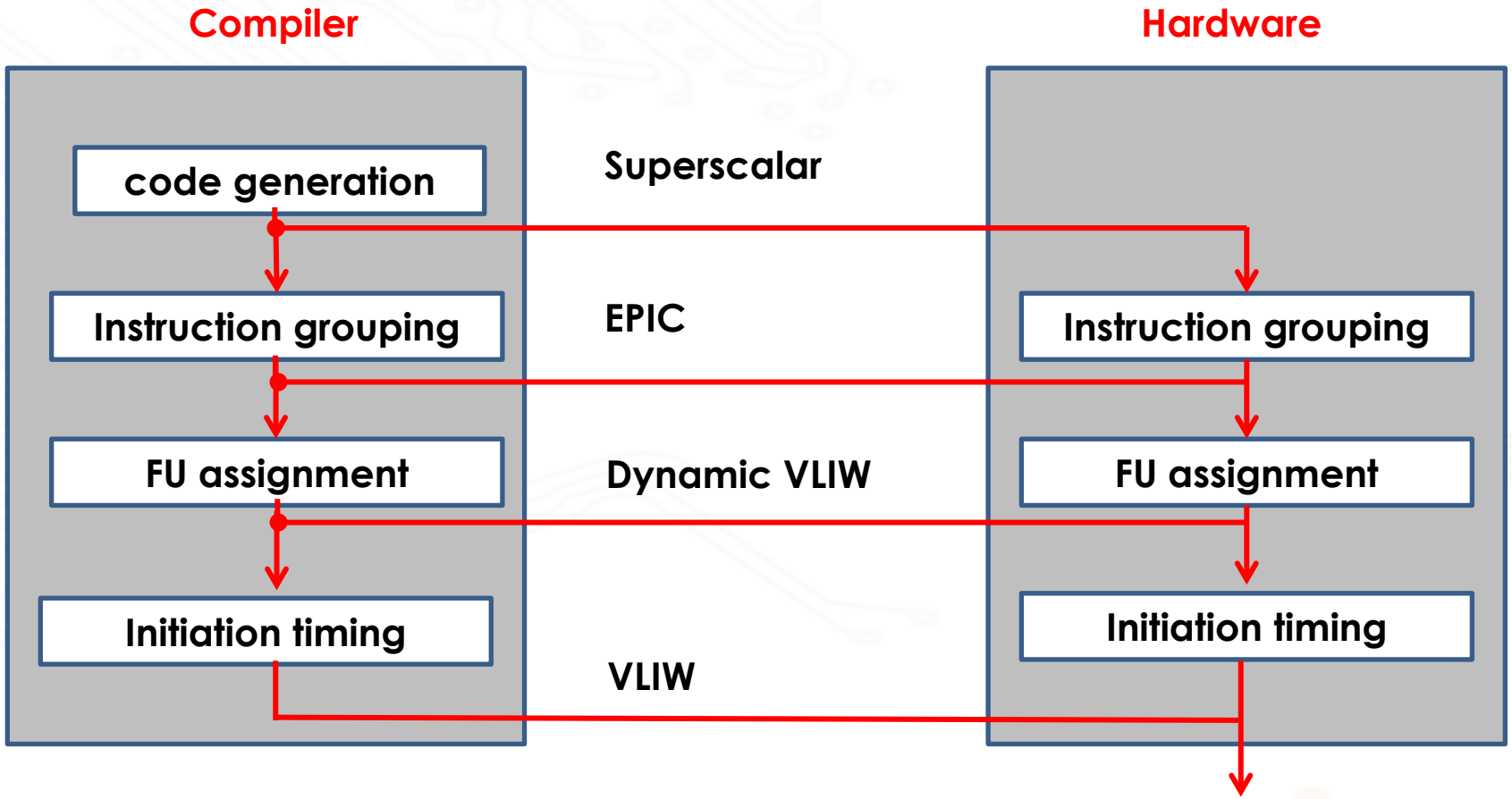
Instruction Level Parallelism and Multi-issue processors

- Execution of more than one instruction at the same time **(in parallel)** => **Instruction Level Parallelism (ILP)**
- **Multi-issue datapath** => Datapath that allow execution of two or more instructions in parallel
- ***N*-issue architecture** => Executes '***N***' instructions at the same time



Multi-issue/super scalar processor =>2 or more instructions are issued in a clock cycle

Hardware and Software Approaches



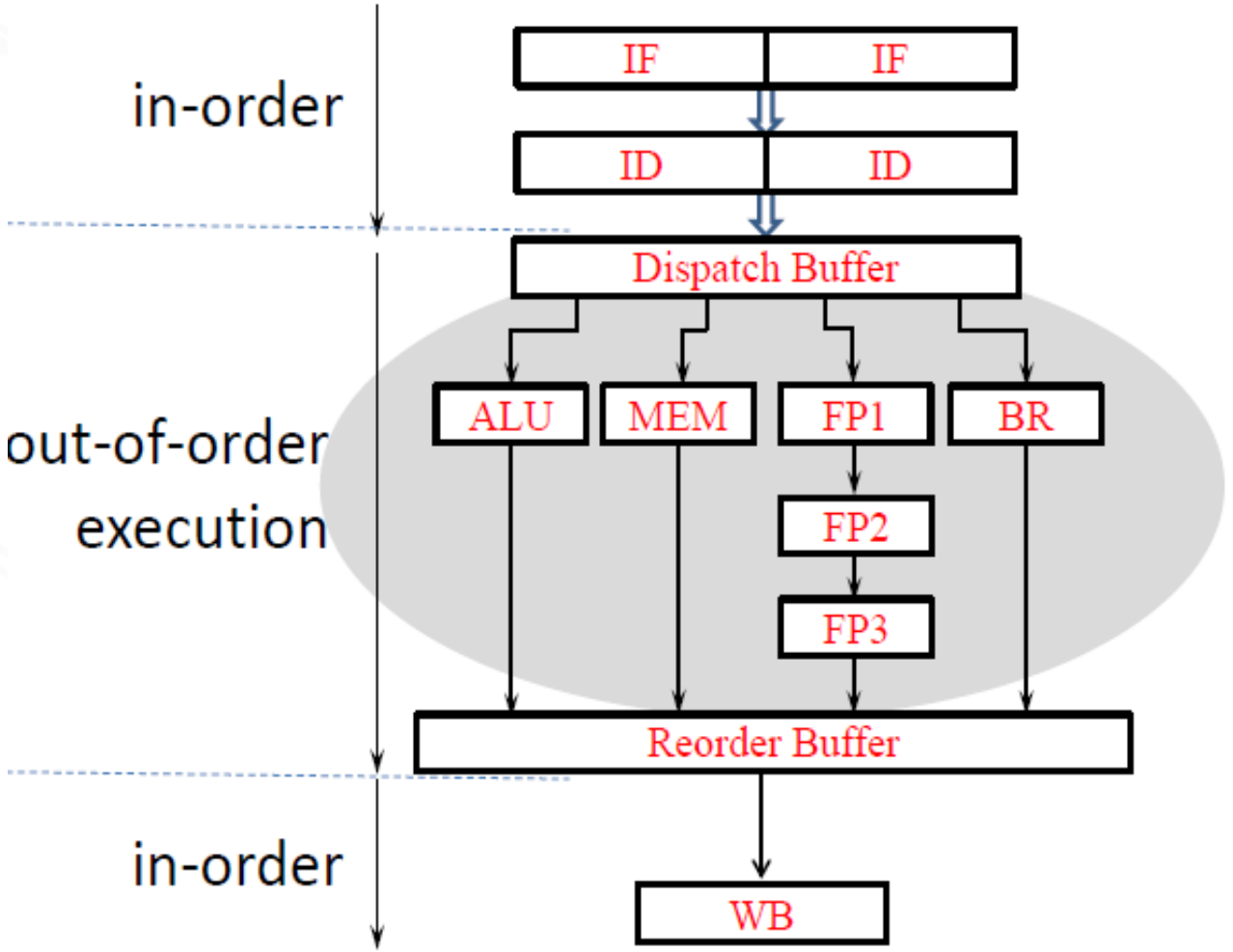
FU: Functional unit

EPIC: Explicitly parallel instruction computing

- Hardware Approach: **Superscalar** Processor
- Software (or compiler-based) Approach: **Very Long Instruction Word(VLIW)** processor

Out-of-order execution in SS pipeline

- Multiple Instructions are fetched in compiler-generated order **(in-order).**
- Instruction completion also **in order**.
- Instructions are executed in some other order: **independent instructions behind a stalled instruction can be executed prior to the stalled instruction.**
- Dynamically scheduled: the order of execution of instructions is changed.



Methods to extract more parallelism

1. **Instruction reordering and out-of-order execution,**
2. **Speculative execution with dynamic scheduling,**
3. **Loop unrolling**

- Instruction Reordering: Change the order of execution of instruction if it does not violate the data dependence.
- Speculative Execution: To execute a instruction without exactly knowing if that need to be executed: ahead of branch outcome.
- Dynamic Scheduling: Execute instructions as soon as dependencies are satisfied and functional units are available.

Loop unrolling for ILP

- Loop unrolling can be used to reduce the total number of instructions (as you saw in Q2 of tutorial 2)
- It can be used to reduce the CPI by unrolling and instruction reordering. (shown in the pipeline example slides)
- Here in Super scaler processors our intention is to have as many independent instructions to do efficient multi issue.

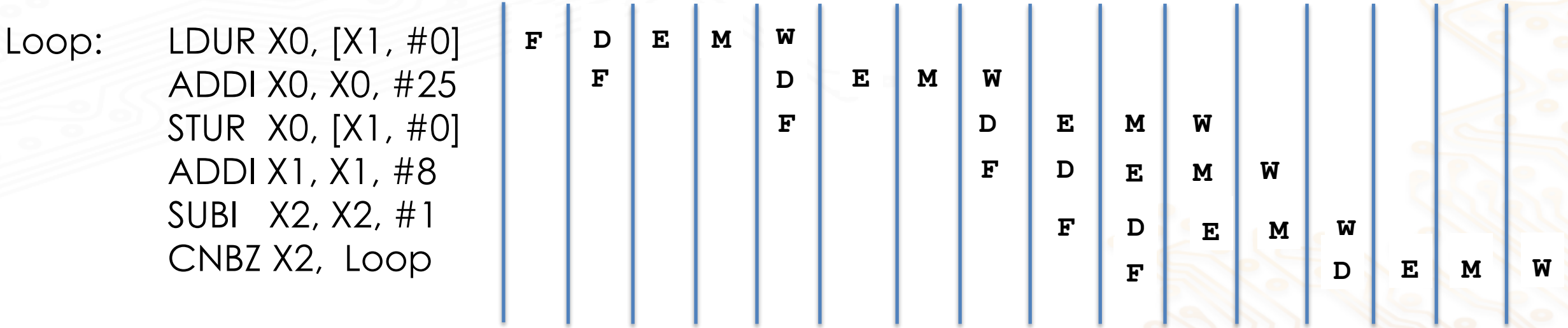
How can loop unrolling help us to produce multiple independent instructions?

Example: Scheduling for superscalar

Dec and WB simultaneously

```
Loop:   LDUR X0, [X1, #0]    # load an array element
        ADDI X0, X0, #25     # add that with 25
        STUR  X0, [X1, #0]   # store back the result
        ADDI X1, X1, #8      # find address of the next element in dmem
        SUBI   X2, X2, #1    # find the remaining no. of iterations
        CNBZ X2,  Loop        # continue iterations if x2 != 0
```

Control hazard is removed in decode stage (1 stall cycle)



Example: Scheduling for superscalar

```
Loop:   LDUR X0, [X1, #0]
        ADDI X0, X0, #25
        STUR  X0, [X1, #0]
        ADDI X1, X1, #8
        SUBI   X2, X2, #1
        CBNZ X2,  Loop
```

(2 stalls)
(2 stalls)
(2 stalls)
(1 stall)

Control hazard is removed in decode stage (1 stall cycle)
CPI = (6+7)/6=2.17 without reordering in a **scalar processor**

2-way super-scalar after instruction reordering
CPI = 7/6 (No data-forwarding is done)

	Way-1 (rest of instructions)	Way-2 (LDUR and STUR only)	Cycle
Loop	ADDI X1, X1, #8	LDUR X0, [X1, #0]	1
	SUBI X2, X2, #1	nop	2
	nop	nop	3
	ADDI X0, X0, #25	nop	4
	nop	nop	5
	CBNZ X2, Loop	nop	6
	CBNZ X2, Loop	STUR X0, [X1, #-8]	7

Another way

```
Loop:   LDUR X0, [X1, #0]
        ADDI X0, X0, #25      (2 stalls)
        STUR  X0, [X1, #0]    (2 stalls)
        ADDI X1, X1, #8
        SUBI   X2, X2, #1     (2 stalls)
        CBNZ X2,  Loop        (1 stall)
```

(No data-forwarding is done (DEC and WB together)

	Way-1 (rest of instructions)	Way-2 (LDUR and STUR only)	Cycle
Loop	SUBI X2, X2, #1	LDUR X0, [X1, #0]	1
	nop	nop	2
	nop	nop	3
	ADDI X0, X0, #25	nop	4
	ADDI X1, X1, #8	nop	5
	CBNZ X2, Loop	nop	6
	CBNZ X2, Loop	STUR X0, [X1, #0]	7

Loop unrolling and instruction reordering

```
Loop: LDUR X0, [X1, #0]
      ADDI X0, X0, #25
      STUR  X0, [X1, #0]
      ADDI X1, X1, #8
      SUBI  X2, X2, #1
      CBNZ X2,  Loop
```

```
Loop: LDUR X0, [X1, #0]
      ADDI X0, X0, #25
      STUR X0, [X1, #0]
      LDUR X3, [X1, #8]
      ADDI X3, X3, #25
      STUR X3, [X1, #8]
      LDUR X4, [X1, #16]
      ADDI X4, X4, #25
      STUR X4, [X1, #16]
      LDUR X5, [X1, #24]
      ADDI X5, X5, #25
      STUR X5, [X1, #24]
      ADDI X1,X1, #32
      SUBI X2, X2, #4
      CBNZ X2,  Loop
```

```
Loop: LDUR X0, [X1, #0]
      LDUR X3, [X1, #8]
      LDUR X4, [X1, #16]
      LDUR X5, [X1, #24]
      ADDI X0, X0, #25
      ADDI X3, X3, #25
      ADDI X4, X4, #25
      ADDI X5, X5, #25
      SUBI X2, X2, #4
      STUR X0, [X1, #0]
      STUR X3, [X1, #8]
      STUR X4, [X1, #16]
      STUR X5, [X1, #24]
      ADDI X1,X1, #32
      CBNZ X2,  Loop
```

after loop unrolling by 4 and instruction reordering

Loop unrolling and instruction reordering

```
Loop:   LDUR X0, [X1, #0]
        LDUR X3, [X1, #8]
        LDUR X4, [X1, #16]
        LDUR X5, [X1, #24]
        ADDI X0, X0, #25
        ADDI X3, X3, #25
        ADDI X4, X4, #25
        ADDI X5, X5, #25
        SUBI X2, X2, #4
        STUR X0, [X1, #0]
        STUR X3, [X1, #8]
        STUR X4, [X1, #16]
        STUR X5, [X1, #24]
        ADDI X1,X1, #32
        CBNZ X2,  Loop
```

	Way-1	Way-2	Cycle
Loop	SUBI X2, X2, #4	LDUR X0, [X1, #0]	1
	nop	LDUR X3, [X1, #8]	2
	nop	LDUR X4, [X1, #16]	3
	ADDI X0, X0, #25	LDUR X5, [X1, #24]	4
	ADDI X3, X3, #25	nop	5
	ADDI X4, X4, #25	nop	6
	ADDI X5, X5, #25	STUR X0, [X1, #0]	7
	ADDI X1,X1, #32	STUR X3, [X1, #8]	8
	CBNZ X2, Loop	STUR X4, [X1, # 16]	9
	nop	STUR X5, [X1, #24]	10

2-way super-scalar after 4-unrolling and reordering : CPI = 10/15= 2/3

Old exam paper question

3. (a) The code segment shown in Listing Q3 is intended to be executed in a two-way superscalar processor. In the superscalar processor, one way is exclusively for load and store instructions and the second way can execute all instructions except load and store. Assume that write-back and register-read operations of different instructions can be performed in the same clock cycle and there is no data forwarding operation. Note that the branch is resolved and updated in the decode stage. The base address of array 'x' is in register X0 and X2 has a value 0x100.

Perform unrolling of the loop in Listing Q3 by a factor of 2 and do the necessary reordering of instructions for reducing the number of stall cycles to the minimum. Compare the change in the CPI achieved by loop unrolling and instruction reordering to the unrolled version. You are allowed to use new temporary registers to get rid of hazards.

Conclusions

- Pipeline data-path
- Challenges in ILP realization
- Data-dependence
- Pipeline hazards and their solutions
- Out-of-order execution
- Branch prediction
- Dynamic scheduling
- VLIW and superscalar processors