The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic look.

Computer Architecture

Advanced Pipelining and Superscalar Processors

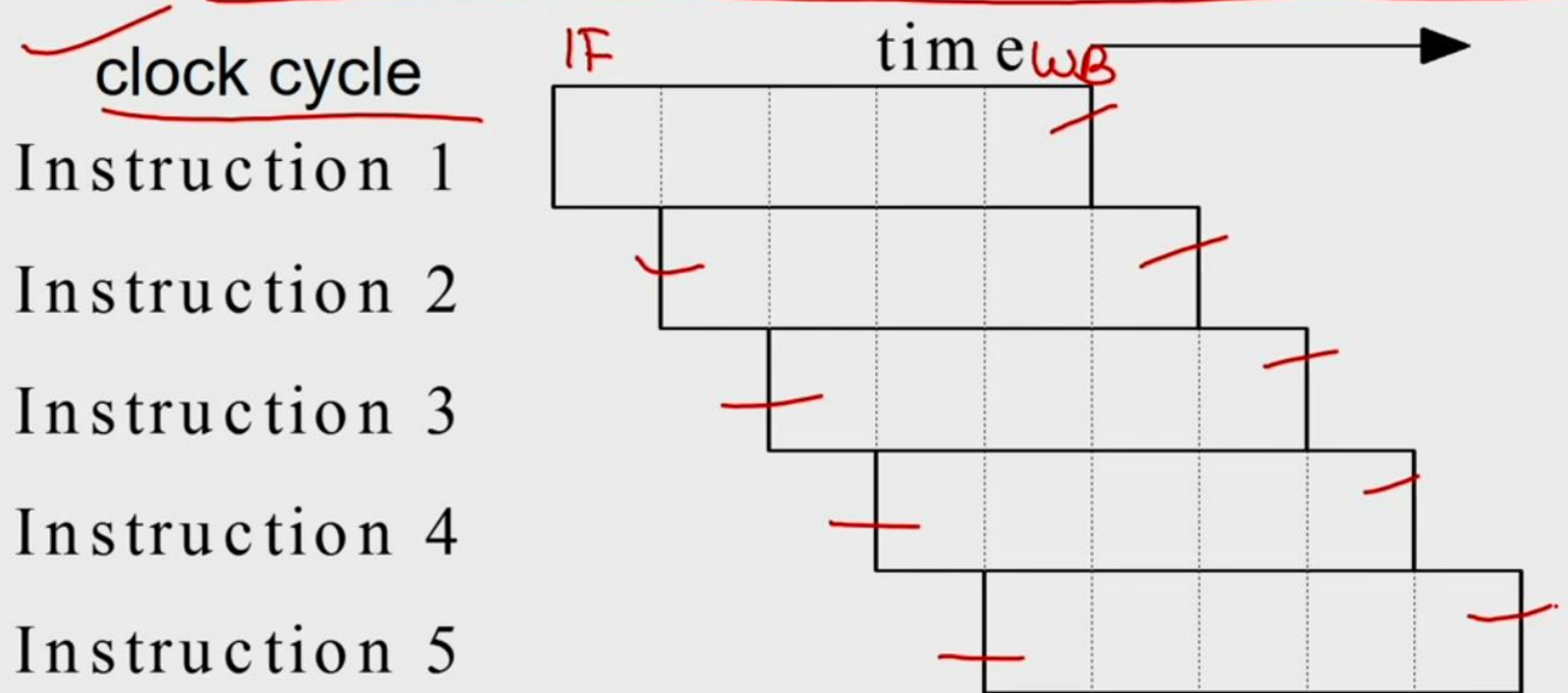
Dr. Mohammad Reza Selim

Lecture Outline

- ▶ Advanced Pipelining Introduction
- ▶ VLIW Processors
- ▶ Symmetric Multi-processor
- ▶ Multi-threading
- ▶ Hyperthreading

Simple In-order Pipeline

❖ pipelining – goal was to complete one instruction per



Advanced Pipelining

- ❖ Increase the depth of the pipeline to increase the clock rate – **superpipelining**
- ❖ Fetch (and execute) more than one instructions at one time (expand every pipeline stage to accommodate multiple instructions) – **multiple-issue (super scalar)**
- ❖ Launching multiple instructions per stage allows the instruction execution rate, CPI, to be less than 1

Superpipelining

❖ **superpipelining** - Increase the depth of the pipeline to

increase the clock rate

time →

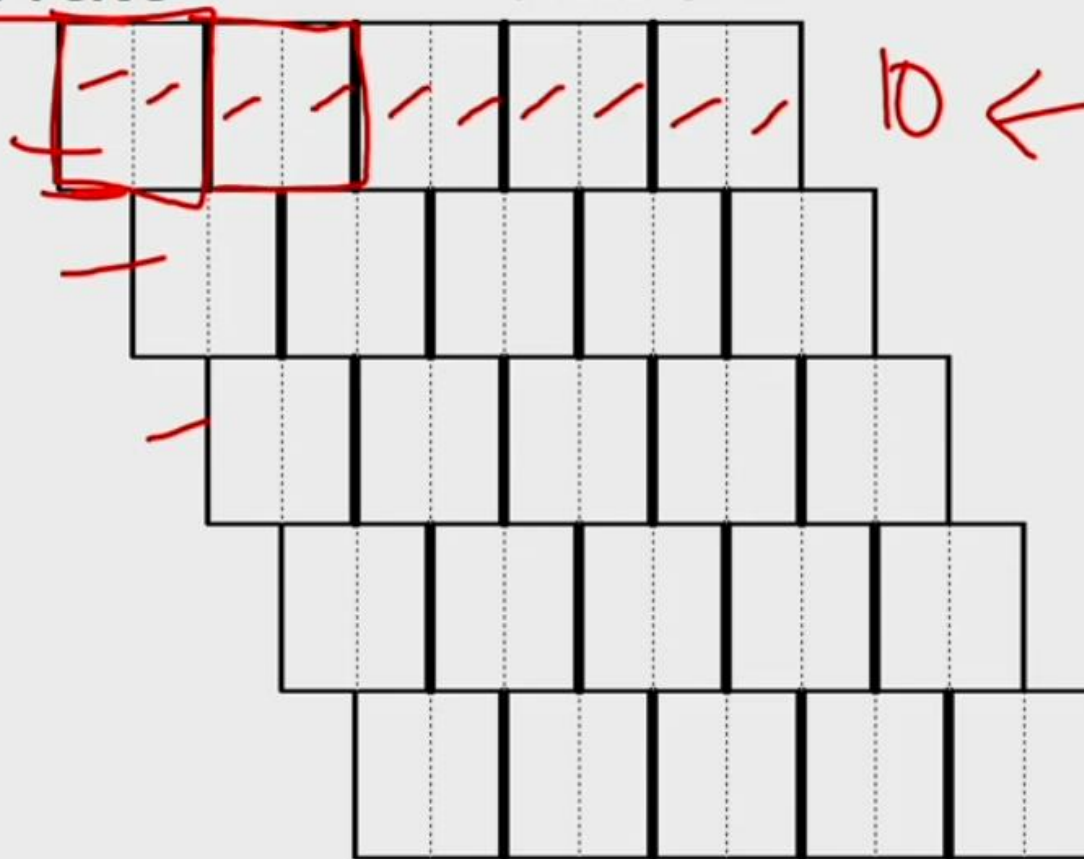
Instruction 1

Instruction 2

Instruction 3

Instruction 4

Instruction 5



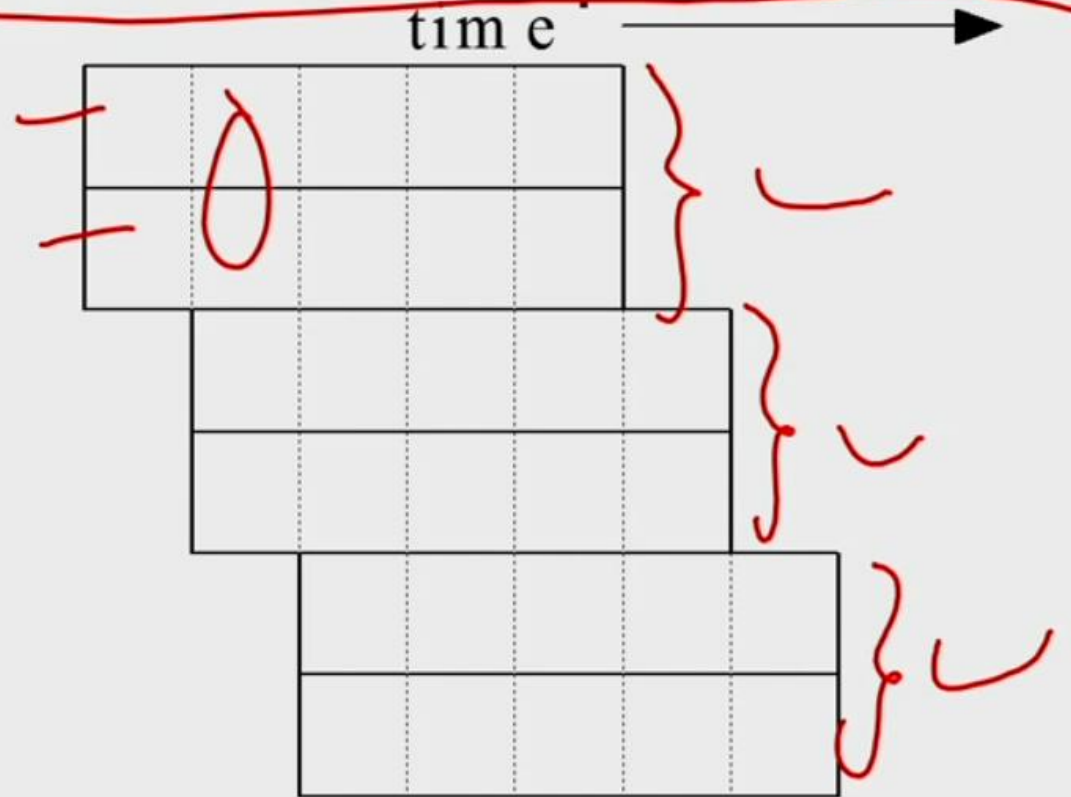
10 ← 5
1ns
0.5ns
//

Superscalar - Multiple Issue

- ❖ Fetch (and execute) more than 1 instructions at one time
- ❖ Expand every stage to accommodate multiple instructions

2
↓
4

Instruction 1
Instruction 2
Instruction 3
Instruction 4
Instruction 5
Instruction 6

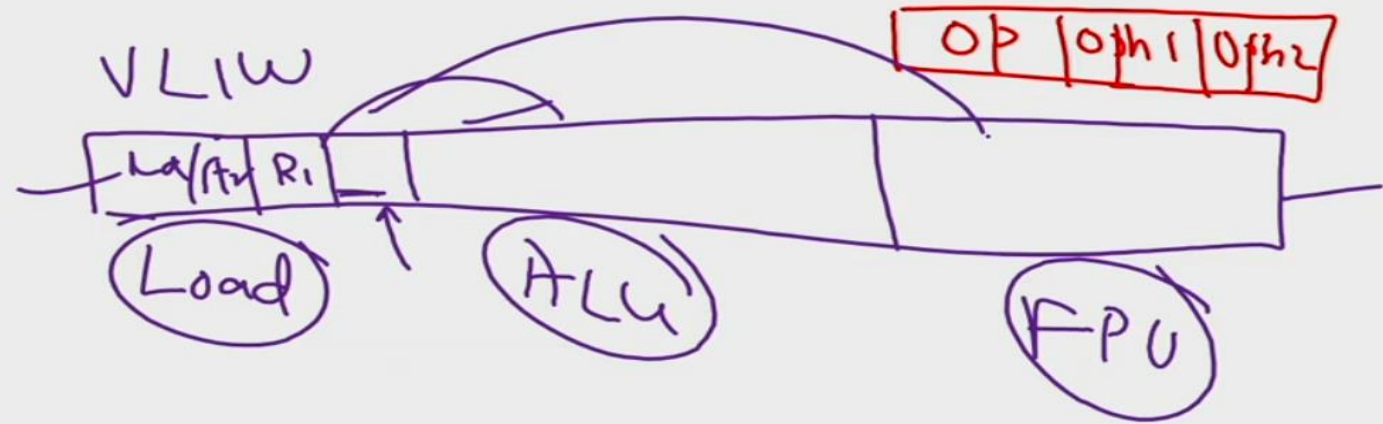


Multiple Issue and Static Scheduling

- ❖ Basic pipeline will give only a min CPI of 1
- ❖ To achieve $CPI < 1$, need to complete multiple instructions per clock (**superscalar processors**)
- ❖ Multiple functional units and Multiple Issue
- ❖ Solutions:
 - ❖ Statically scheduled superscalar processors *Compiler X*
 - ❖ VLIW (very long instruction word) processors
 - ❖ Dynamically scheduled superscalar processors

VLIW Processors

- ❖ Package multiple operations into one instruction (long word)



- ❖ The instruction have a set of fields for each functional unit.
- ❖ 16 to 24 bits per unit, → instruction length of 80 to 120 bits.
- ❖ Must be enough parallelism in code to fill the available slots
- ❖ Find enough parallel instructions by loop unrolling and scheduling

VLIW Processors

- ❖ Disadvantages:
 - ❖ Statically find parallelism
 - ❖ Lack of parallelism –slot waste
 - ❖ Code size increase due to unrolling
 - ❖ No hazard detection hardware

VLIW Processors

❖ Example VLIW processor:

- ❖ One integer instruction (or branch)
- ❖ Two independent floating-point operations
- ❖ Two independent memory references

				Clock cycle issued
Loop:	L.D	F0,0(R1)		1
	<i>stall</i>			2
	ADD.D	F4,F0,F2		3
	<i>stall</i>			4
	<i>stall</i>			5
	S.D	F4,0(R1)		6
	DADDUI	R1,R1,#-8		7
	<i>stall</i>			8
	BNE	R1,R2,Loop		9

Memory reference 1	Memory reference 2	FP operation 1	FP operation 2	Integer operation/branch
L.D F0,0(R1)	L.D F6,-8(R1)			
L.D F10,-16(R1)	L.D F14,-24(R1)			
L.D F18,-32(R1)	L.D F22,-40(R1)	ADD.D F4,F0,F2	ADD.D F8,F6,F2	
L.D F26,-48(R1)		ADD.D F12,F10,F2	ADD.D F16,F14,F2	
		ADD.D F20,F18,F2	ADD.D F24,F22,F2	
S.D F4,0(R1)	S.D F8,-8(R1)	ADD.D F28,F26,F2		
S.D F12,-16(R1)	S.D F16,-24(R1)			DADDUI R1,R1,#-56
S.D F20,24(R1)	S.D F24,16(R1)			
S.D F28,8(R1)				BNE R1,R2,Loop

❖ Performance: 9 cycles to complete 26 instructions. More registers used.

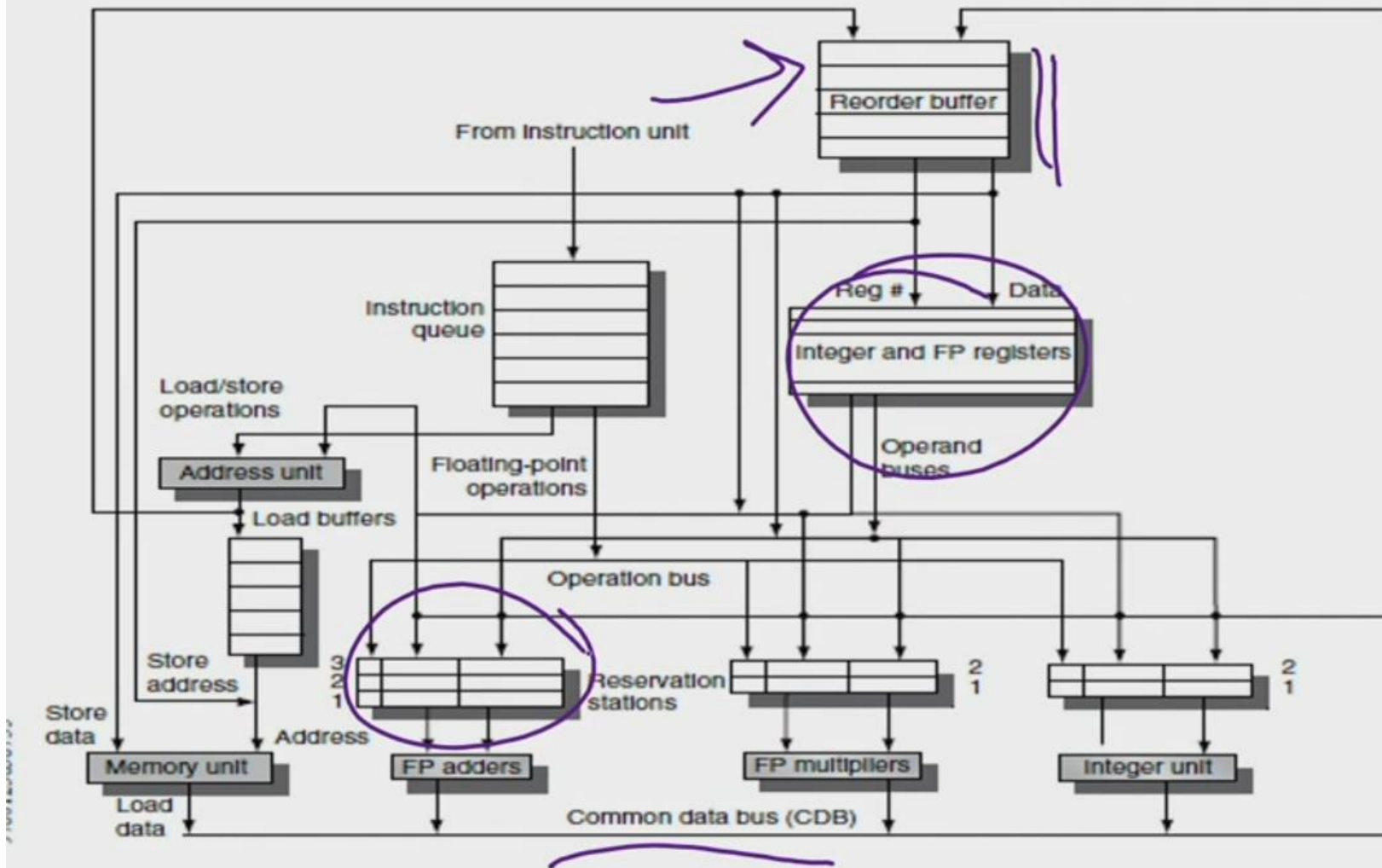
Extreme Optimization

- ❖ Modern micro-architectures uses this triple combination:
 - ❖ Dynamic scheduling + multiple issue + speculation
- ❖ Dependency between instructions issued in same clock.
- ❖ Register read for multiple instructions in parallel.
- ❖ Complex Issue logic to check dependencies and hazards.
- ❖ Assign reservation stations and ROB entries in a cycle.

Handling Multiple Issues

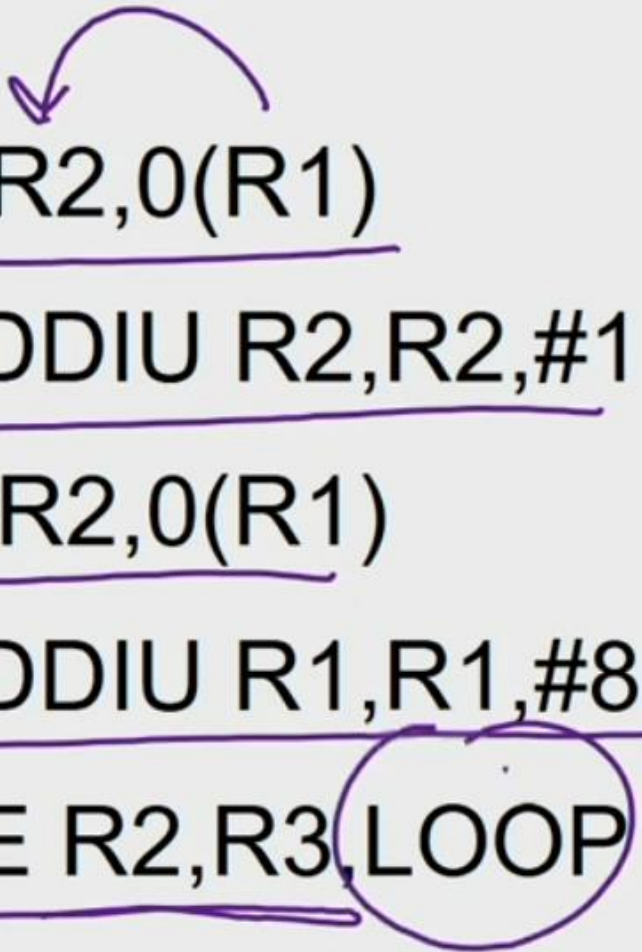
- ❖ Assign RS and ROB for every instruction in next issue bundle.
- ❖ Limit the number of instructions of a given class that can be issued in a bundle[one FP, one integer, one load-store etc]
- ❖ Stall Issue if ROB not available.
- ❖ Examine all the dependencies among the instructions in the bundle
- ❖ If dependencies exist in bundle, encode them in reservation ~~stations~~ with ROB entry.
- ❖ Multiple completion/commit (wide CDB+ROB update)

Handling Multiple Issues



- ❖ One issue per unit per cycle
- ❖ Wider operand buses, CDB
- ❖ Multiple register read/cycle
- ❖ Multiple instruction to commit / cycle

Example



```
Loop: LD R2,0(R1)           ;R2=array element
      DADDIU R2,R2,#1       ;increment R2
      SD R2,0(R1)           ;store result
      DADDIU R1,R1,#8       ;increment pointer
      BNE R2,R3,LOOP       ;branch if not last element
```

Example (Without Speculation)

Iteration number	Instructions	Issues at clock cycle number	Executes at clock cycle number	Memory access at clock cycle number	Write CDB at clock cycle number	Comment
1	LD R2,0(R1)	1	2	3	4	First issue
1	DADDIU R2,R2,#1	1	5		6	Wait for LW
1	SD R2,0(R1)	2	3	7		Wait for DADDIU
1	DADDIU R1,R1,#8	2	3		4	Execute directly
1	BNE R2,R3,LOOP	3	7			Wait for DADDIU
2	LD R2,0(R1)	4	8	9	10	Wait for BNE
2	DADDIU R2,R2,#1	4	11		12	Wait for LW
2	SD R2,0(R1)	5	9	13		Wait for DADDIU
2	DADDIU R1,R1,#8	5	8		9	Wait for BNE
2	BNE R2,R3,LOOP	6	13			Wait for DADDIU
3	LD R2,0(R1)	7	14	15	16	Wait for BNE
3	DADDIU R2,R2,#1	7	17		18	Wait for LW
3	SD R2,0(R1)	8	15	19		Wait for DADDIU
3	DADDIU R1,R1,#8	8	14		15	Wait for BNE
3	BNE R2,R3,LOOP	9	19			Wait for DADDIU

Example (Without Speculation)

Iteration number	Instructions	Issues at clock cycle number	Executes at clock cycle number	Memory access at clock cycle number	Write CDB at clock cycle number	Comment
1	LD R2,0(R1)	1	2	3	4	First issue
1	DADDIU R2,R2,#1	1	5		6	Wait for LW
1	SD R2,0(R1)	2	3	7		Wait for DADDIU
1	DADDIU R1,R1,#8	2	3		4	Execute directly
1	BNE R2,R3,LOOP	3	7			Wait for DADDIU
2	LD R2,0(R1)	4	8	9	10	Wait for BNE
2	DADDIU R2,R2,#1	4	11		12	Wait for LW
2	SD R2,0(R1)	5	9	13		Wait for DADDIU
2	DADDIU R1,R1,#8	5	8		9	Wait for BNE
2	BNE R2,R3,LOOP	6	13			Wait for DADDIU
3	LD R2,0(R1)	7	14	15	16	Wait for BNE
3	DADDIU R2,R2,#1	7	17		18	Wait for LW
3	SD R2,0(R1)	8	15	19		Wait for DADDIU
3	DADDIU R1,R1,#8	8	14		15	Wait for BNE
3	BNE R2,R3,LOOP	9	19			Wait for DADDIU

Example (With Speculation)

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,#8	2	3		4	8	Commit in order
1	<u>BNE R2,R3,LOOP</u>	3	7			8	Wait for DADDIU
2	LD R2,0(R1)	4	5	6	7	9	No execute delay
2	DADDIU R2,R2,#1	4	8		9	10	Wait for LW
2	SD R2,0(R1)	5	6			10	Wait for DADDIU
2	DADDIU R1,R1,#8	5	6		7	11	Commit in order
2	BNE R2,R3,LOOP	6	10			11	Wait for DADDIU
3	LD R2,0(R1)	7	8	9	10	12	Earliest possible
3	DADDIU R2,R2,#1	7	11		12	13	Wait for LW
3	SD R2,0(R1)	8	9			13	Wait for DADDIU
3	DADDIU R1,R1,#8	8	9		10	14	Executes earlier
3	BNE R2,R3,LOOP	9	13			14	Wait for DADDIU

Example (With Speculation)

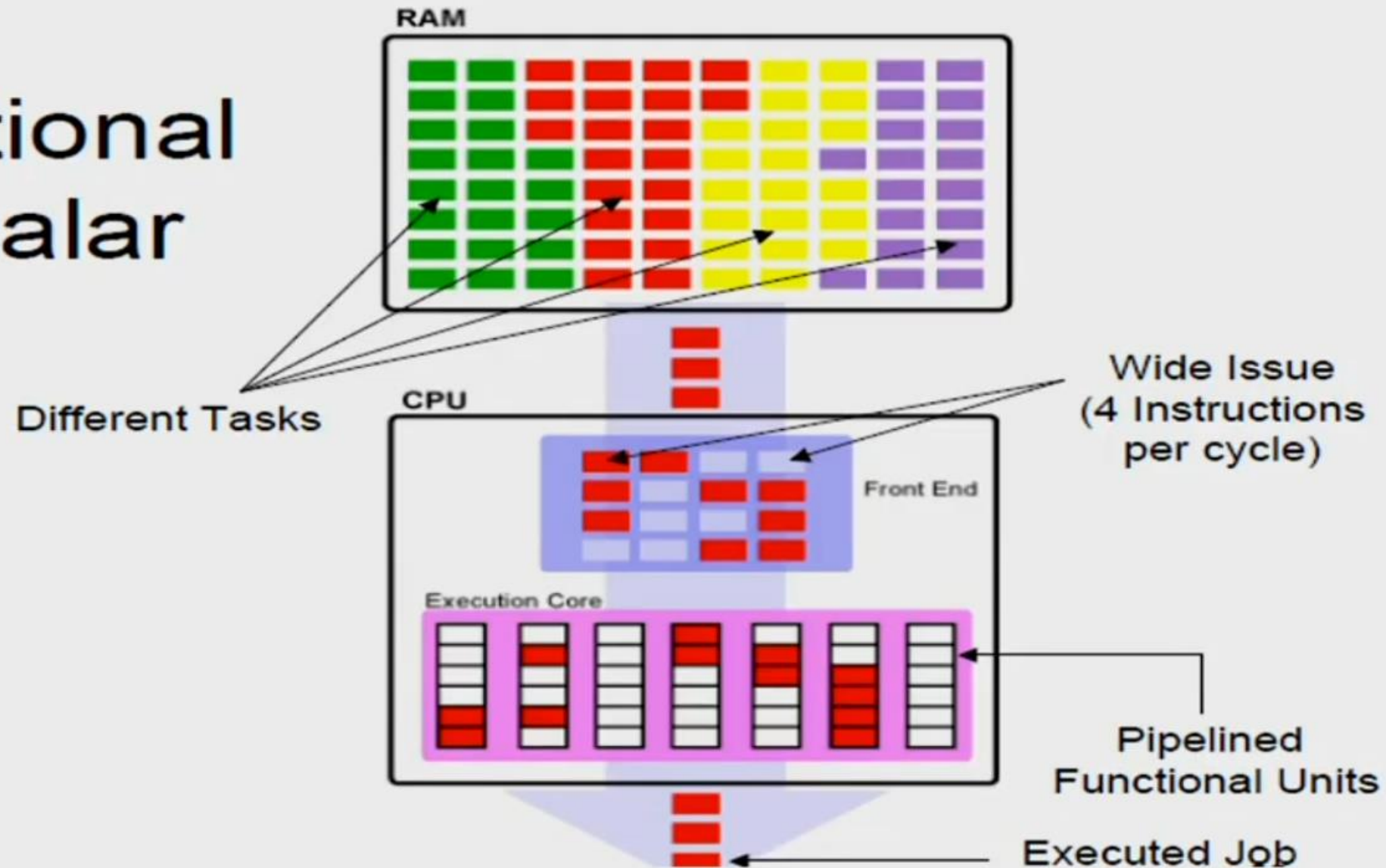
Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,#8	2	3		4	8	Commit in order
1	<u>BNE R2,R3,LOOP</u>	3	7			8	Wait for DADDIU
2	LD R2,0(R1)	4	5	6	7	9	No execute delay
2	DADDIU R2,R2,#1	4	8		9	10	Wait for LW
2	SD R2,0(R1)	5	6			10	Wait for DADDIU
2	DADDIU R1,R1,#8	5	6		7	11	Commit in order
2	BNE R2,R3,LOOP	6	10			11	Wait for DADDIU
3	LD R2,0(R1)	7	8	9	10	12	Earliest possible
3	DADDIU R2,R2,#1	7	11		12	13	Wait for LW
3	SD R2,0(R1)	8	9			13	Wait for DADDIU
3	DADDIU R1,R1,#8	8	9		10	14	Executes earlier
3	BNE R2,R3,LOOP	9	13			14	Wait for DADDIU

Example (With Speculation)

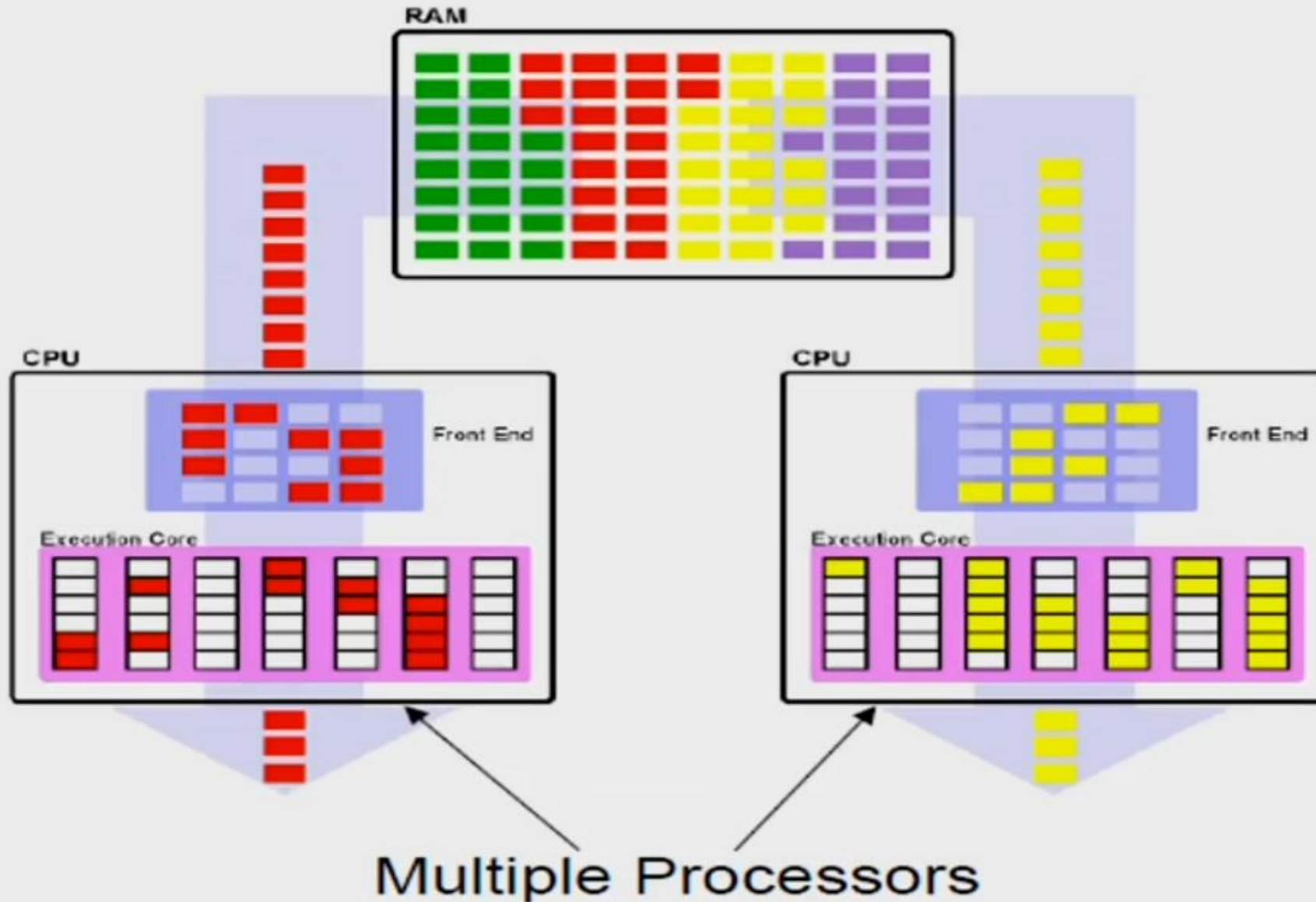
Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,#8	2	3		4	8	Commit in order
1	BNE R2,R3,LOOP	3	7			8	Wait for DADDIU
2	LD R2,0(R1)	4	5	6	7	9	No execute delay
2	DADDIU R2,R2,#1	4	8		9	10	Wait for LW
2	SD R2,0(R1)	5	6			10	Wait for DADDIU
2	DADDIU R1,R1,#8	5	6		7	11	Commit in order
2	BNE R2,R3,LOOP	6	10			11	Wait for DADDIU
3	LD R2,0(R1)	7	8	9	10	12	Earliest possible
3	DADDIU R2,R2,#1	7	11		12	13	Wait for LW
3	SD R2,0(R1)	8	9			13	Wait for DADDIU
3	DADDIU R1,R1,#8	8	9		10	14	Executes earlier
3	BNE R2,R3,LOOP	9	13			14	Wait for DADDIU

Superscalar Processor

Conventional Superscalar



Symmetric Multiprocessor

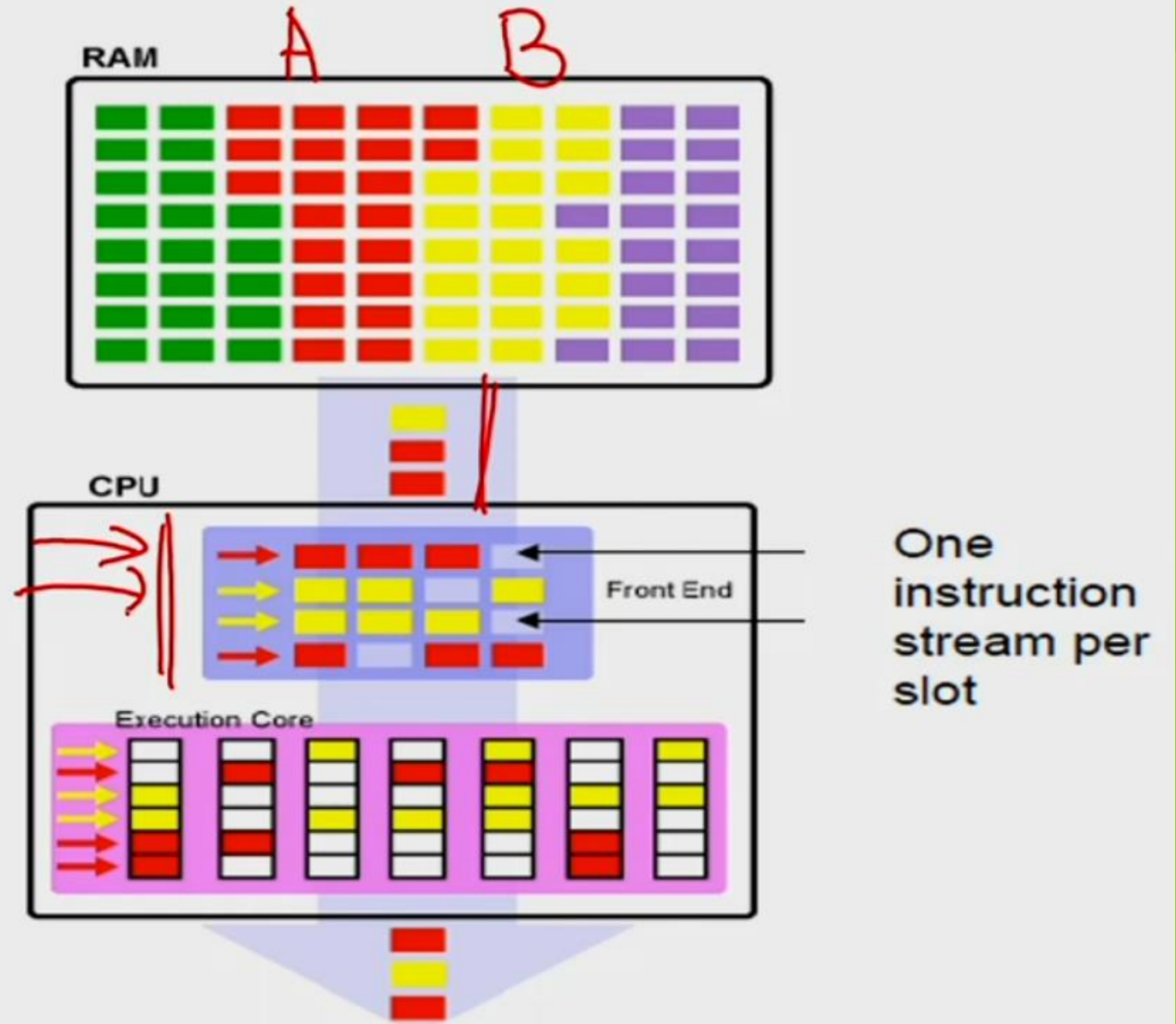


Is that Enough?

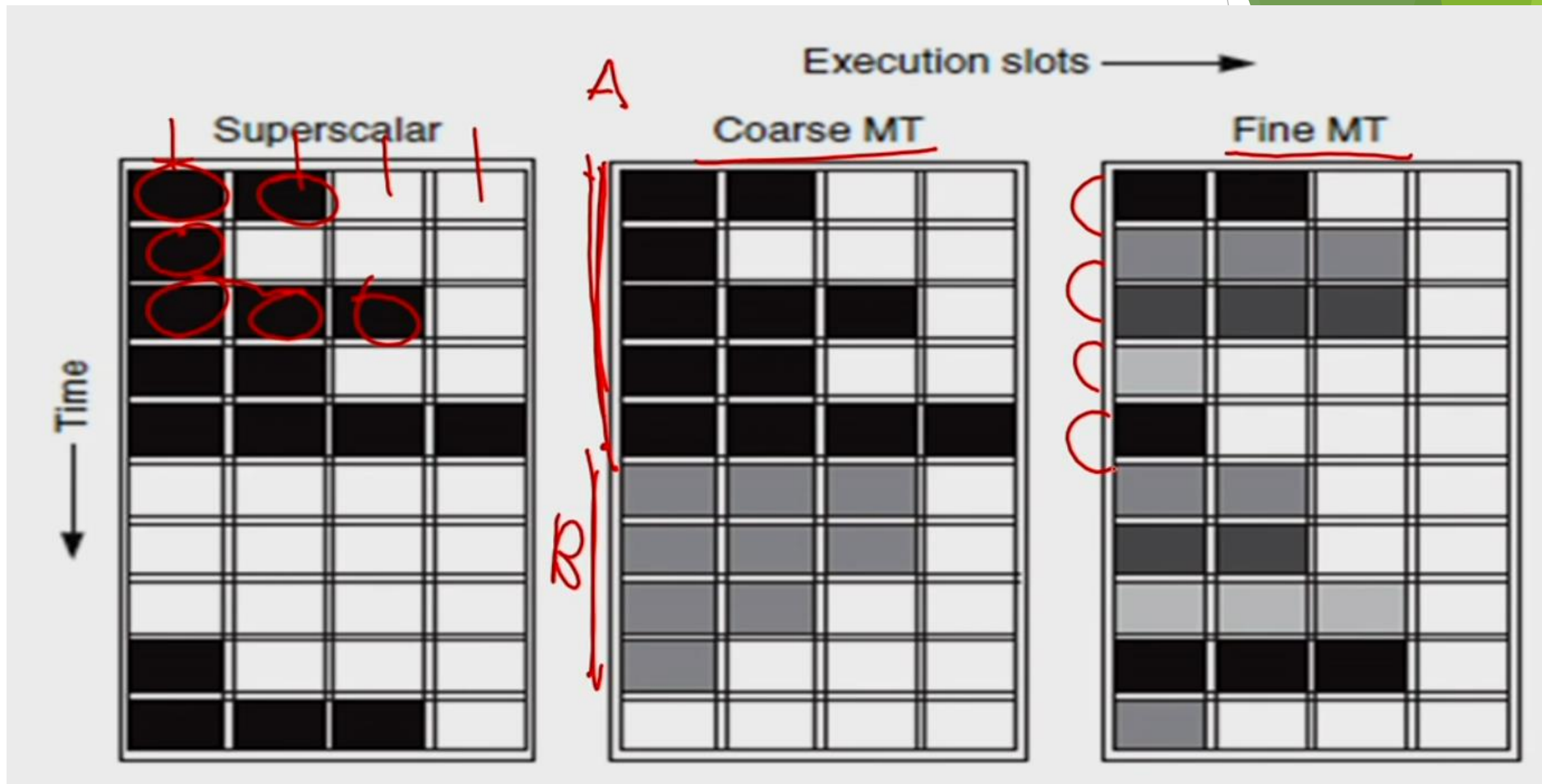
- ❖ We have tried Tomasulo's superscalar approach with ROB, speculation with aggressive branch prediction and multi issue
- ❖ At very high issue rates cache misses that go to L2 and off chip can not be hidden by ILP.
- ❖ How to cover such long memory stalls ?
- ❖ ~~Multithreading~~ allows multiple threads to share the functional units of a single processor in an overlapping fashion

Multithreading

Multithreading

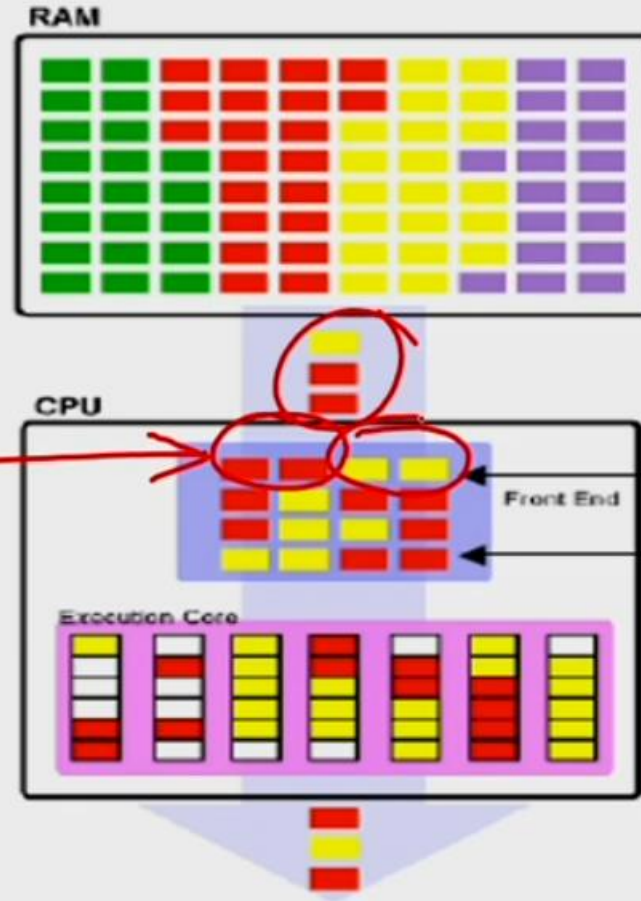


Multithreading



Hyperthreading /SMT

Hyperthreading



More than one
instruction stream
per slot

Comparison of Resource Utilization

