CS222:
Processor Design: Multi-Cycle Design

Dr. A. Sahu

Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

Outline

- Previous:
  - Control Path Implementation
- Multi-cycle Design
  - Problems with single cycle datapath
  - Analyzing delays in single cycle datapath
  - Clock periods in single cycle and multi-cycle designs
  - Improving resource utilization
  - Add registers and multiplexers

MIPS subset for implementation

- Arithmetic - logic instructions
  - **add, sub, and, or, slt**
- Memory reference instructions
  - **lw, sw**
- Control flow instructions
  - **beq, j**

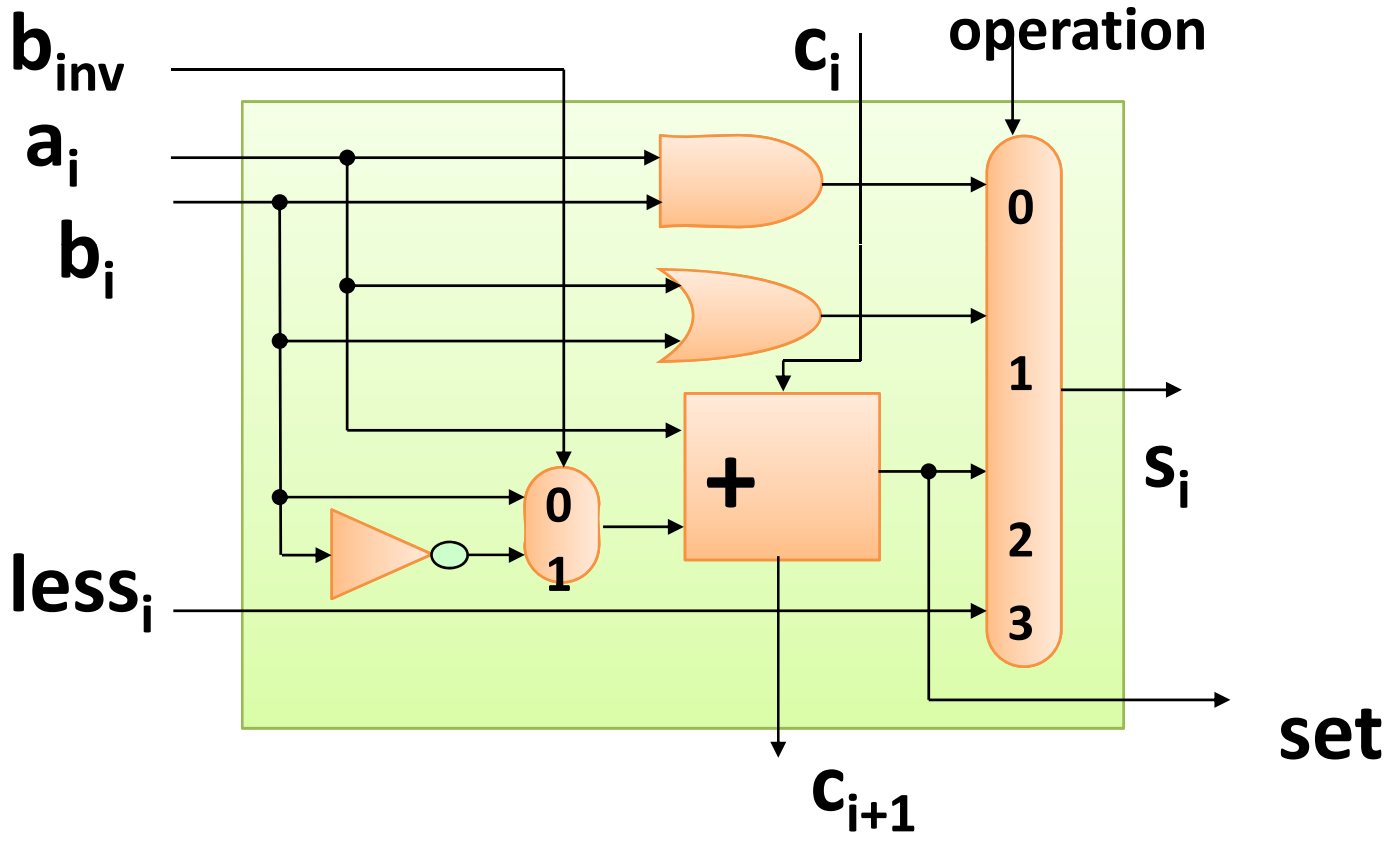
All control inputs

Instruction	Opcode	Rdst	RW	Asrc	MW	MR	M2R	Brn	Jmp
Rtype	000000	1	1	0	0	0	0	0	0
Sw	101011	X	0	1	1	0	X	0	0
Lw	100011	0	1	1	0	1	1	0	0
Beq	000100	X	0	0	0	0	X	1	0
J	000010	X	0	X	0	0	X	X	1

Encoding opc

Ins	Opc ode	OPC	Rdst	RW	Asrc	MW	MR	M2R	Brn	Jmp
R	000000	10	1	1	0	0	0	0	0	0
Sw	101011	00	X	0	1	1	0	X	0	0
Lw	100011	00	0	1	1	0	1	1	0	0
Beq	000100	01	X	0	0	0	0	X	1	0
J	000010	XX	X	0	X	0	0	X	X	1

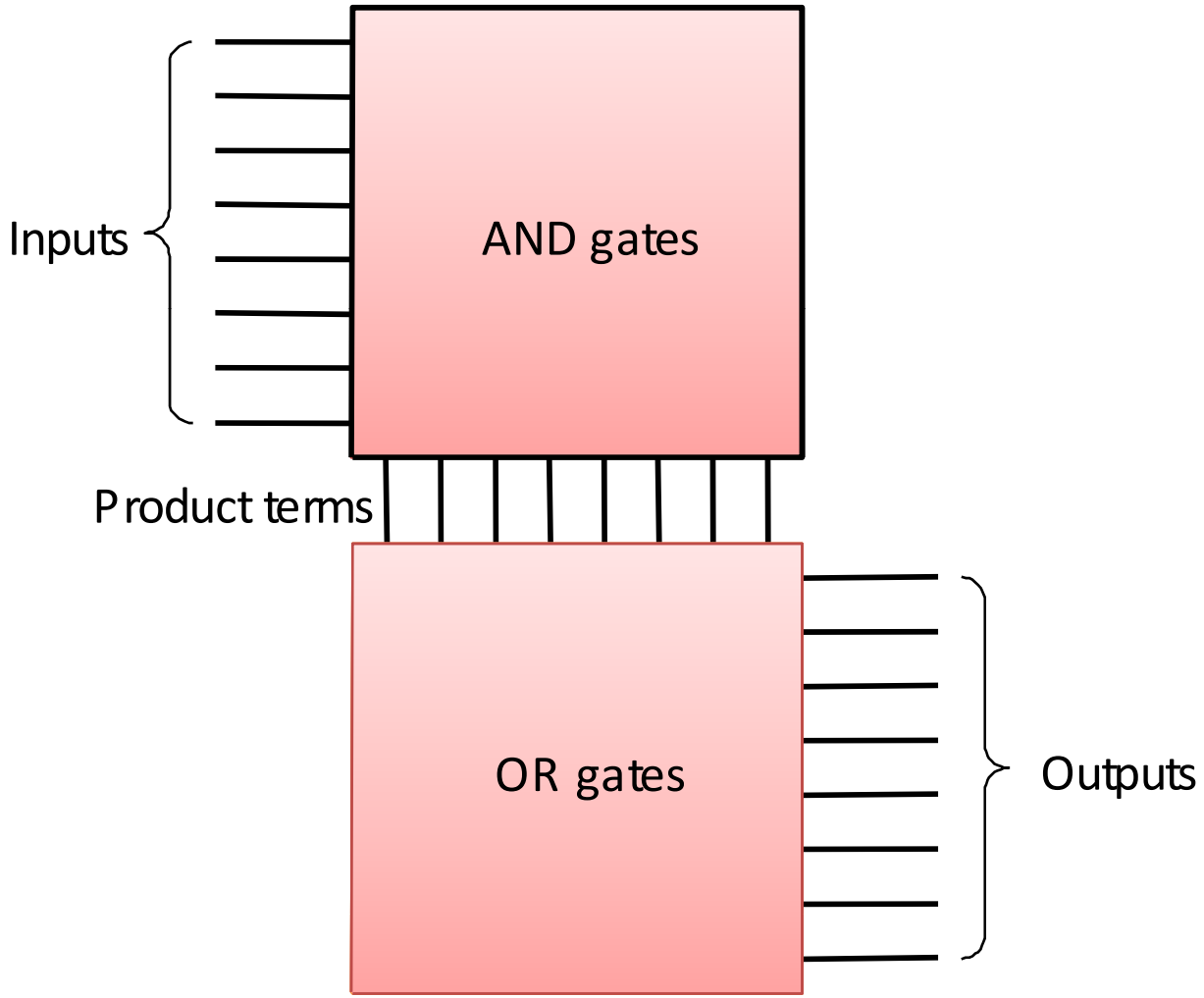
ALU



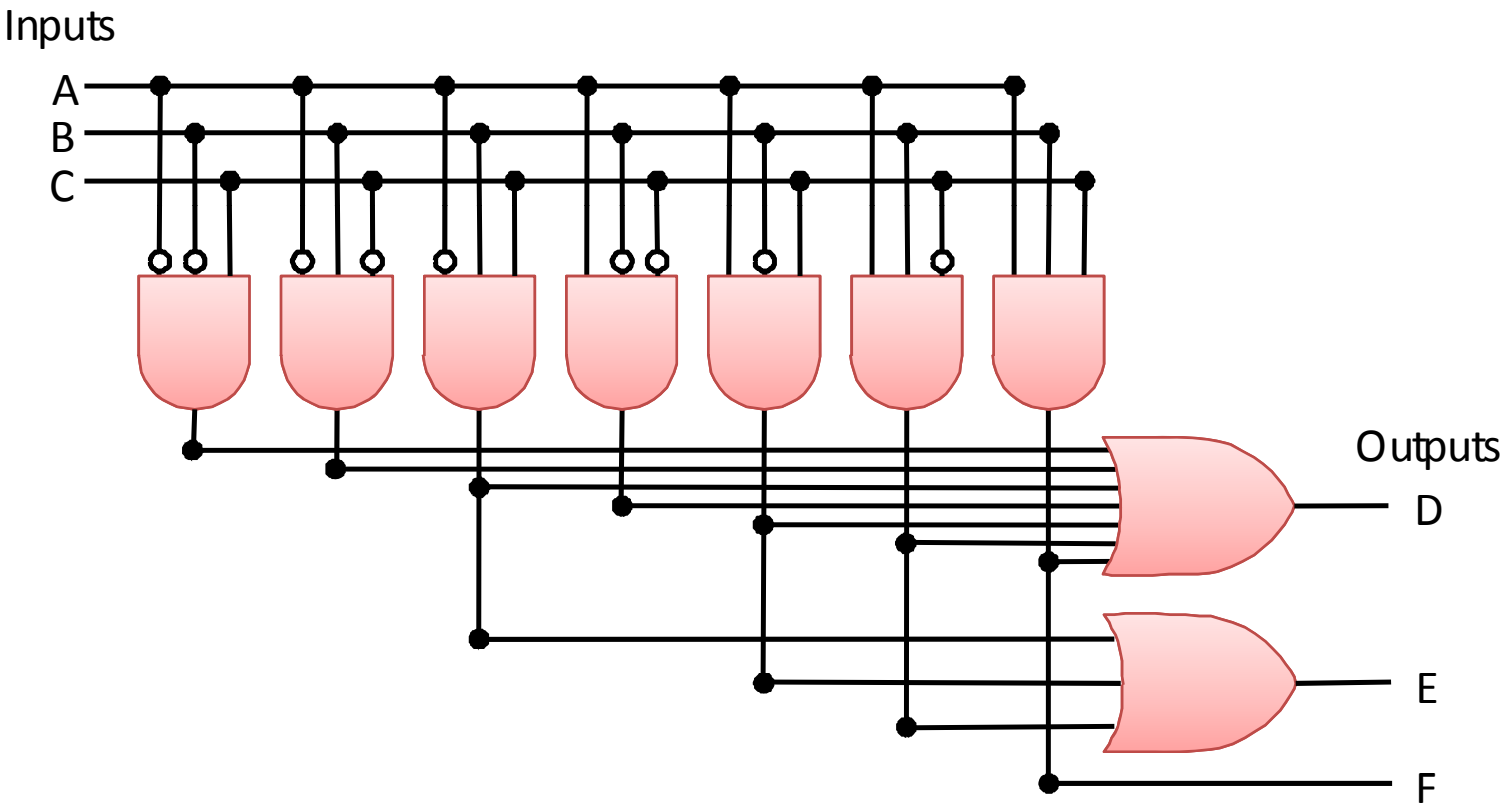
ALU control input

Type	OPC	INS	FUN	Action	OP
R-Type	10	Add	100000	Add	010
R-Type	10	Sub	100010	Sub	110
R-Type	10	And	100100	And	000
R-Type	10	Or	100101	Or	001
R-Type	10	SLT	101010	SetOnLess	111
Sw	00	Sw	Xxxxxx	Add	010
Lw	00	Lw	Xxxxxx	Add	010
Beq	01	Beq	Xxxxxx	Sub	110
J	XX	J	Xxxxxx	Xxx	xxx

Implementing controllers



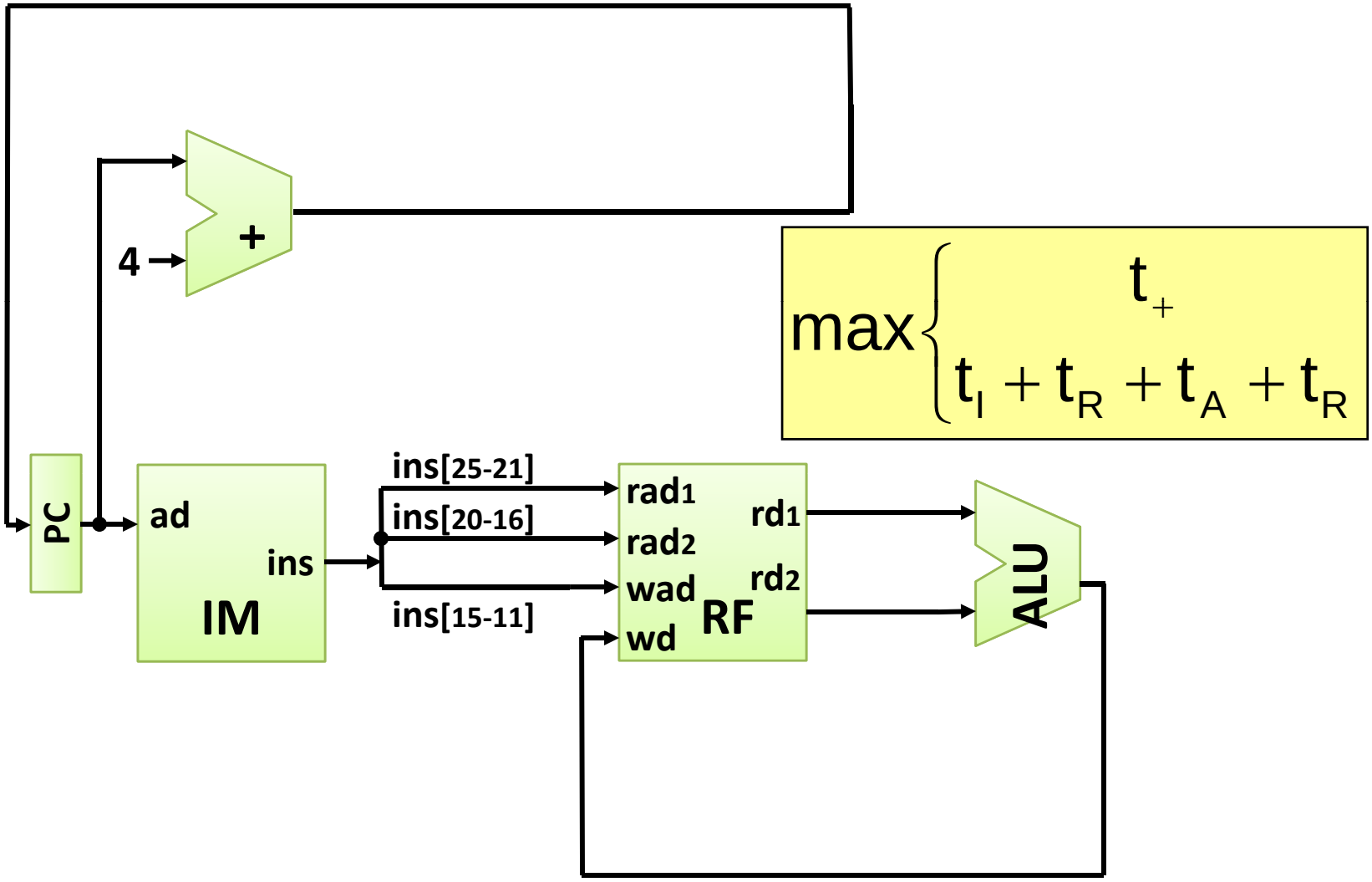
AND-OR PLA



Revisit: Analyzing performance

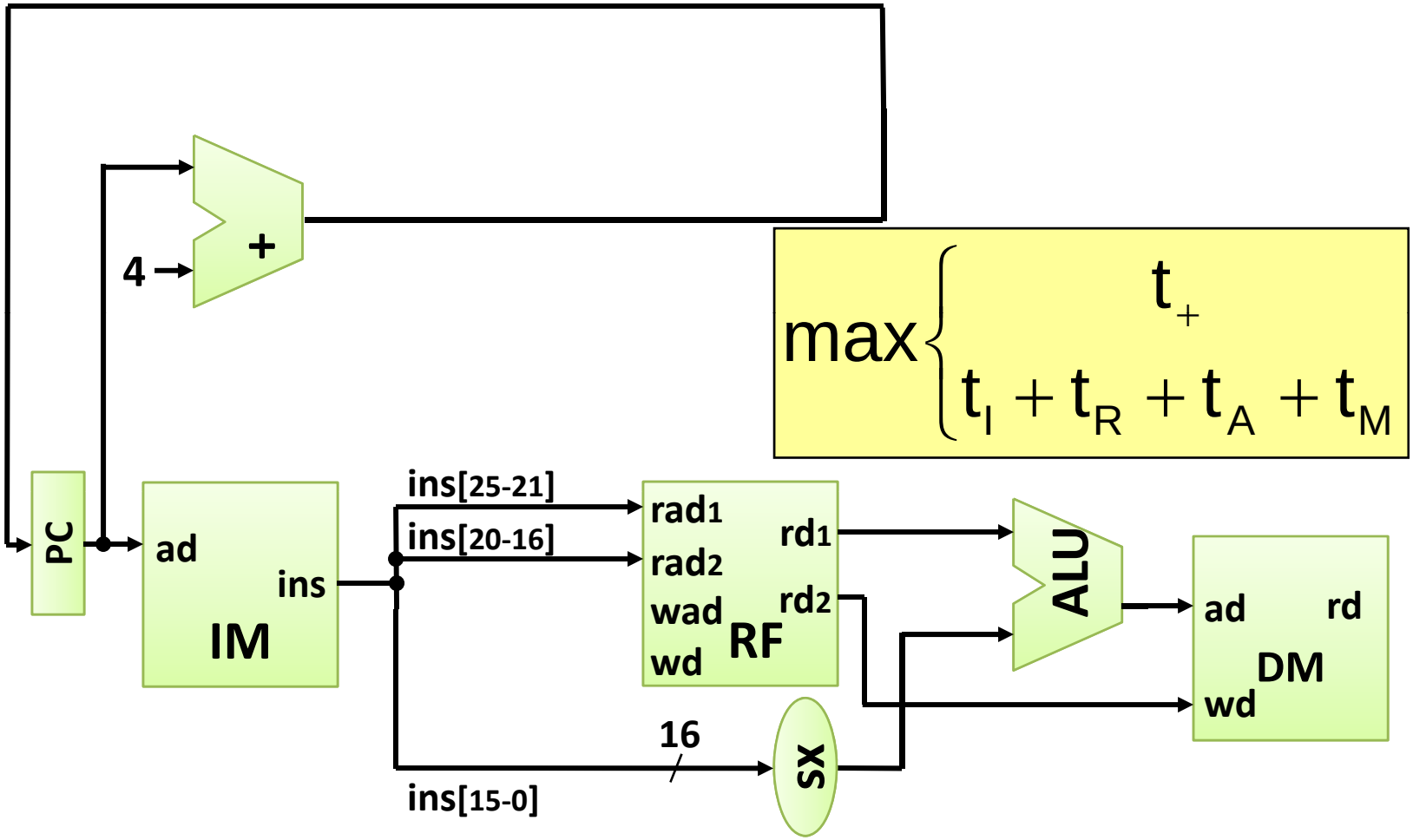
Components	Delay
Register	0
Adder	t_+
ALU	t_A
MUX	0
RF	t_R
Instruction Memory	t_I
Data Memory	t_m
Bit manipulation	0

Delay for {add, sub, and, or, slt}



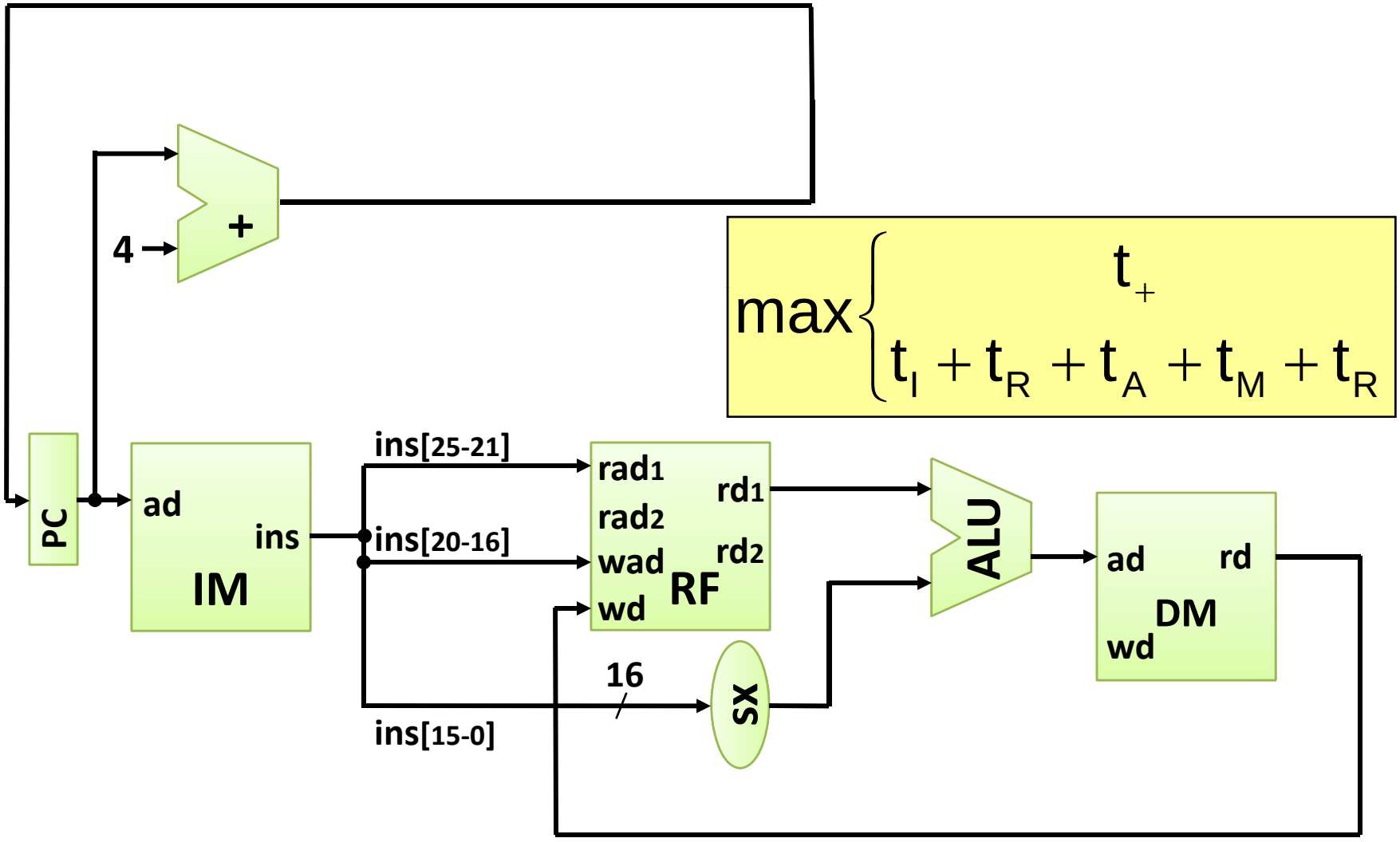
$$\max\begin{cases} t_{+} \\ t_I + t_R + t_A + t_R \end{cases}$$

Delay for {sw}



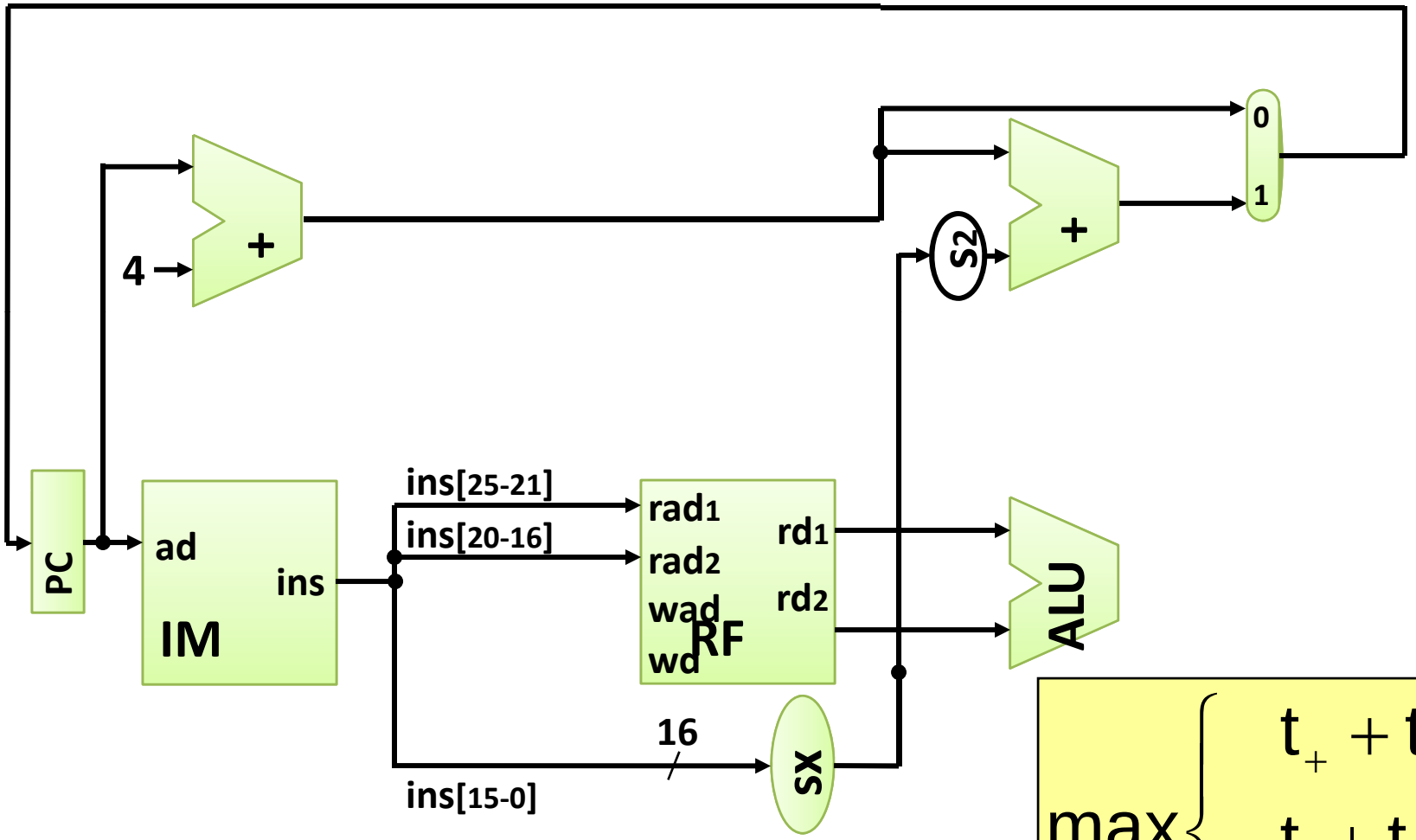
$$\max\begin{cases} t_{+} \\ t_I + t_R + t_A + t_M \end{cases}$$

Delay for {lw}



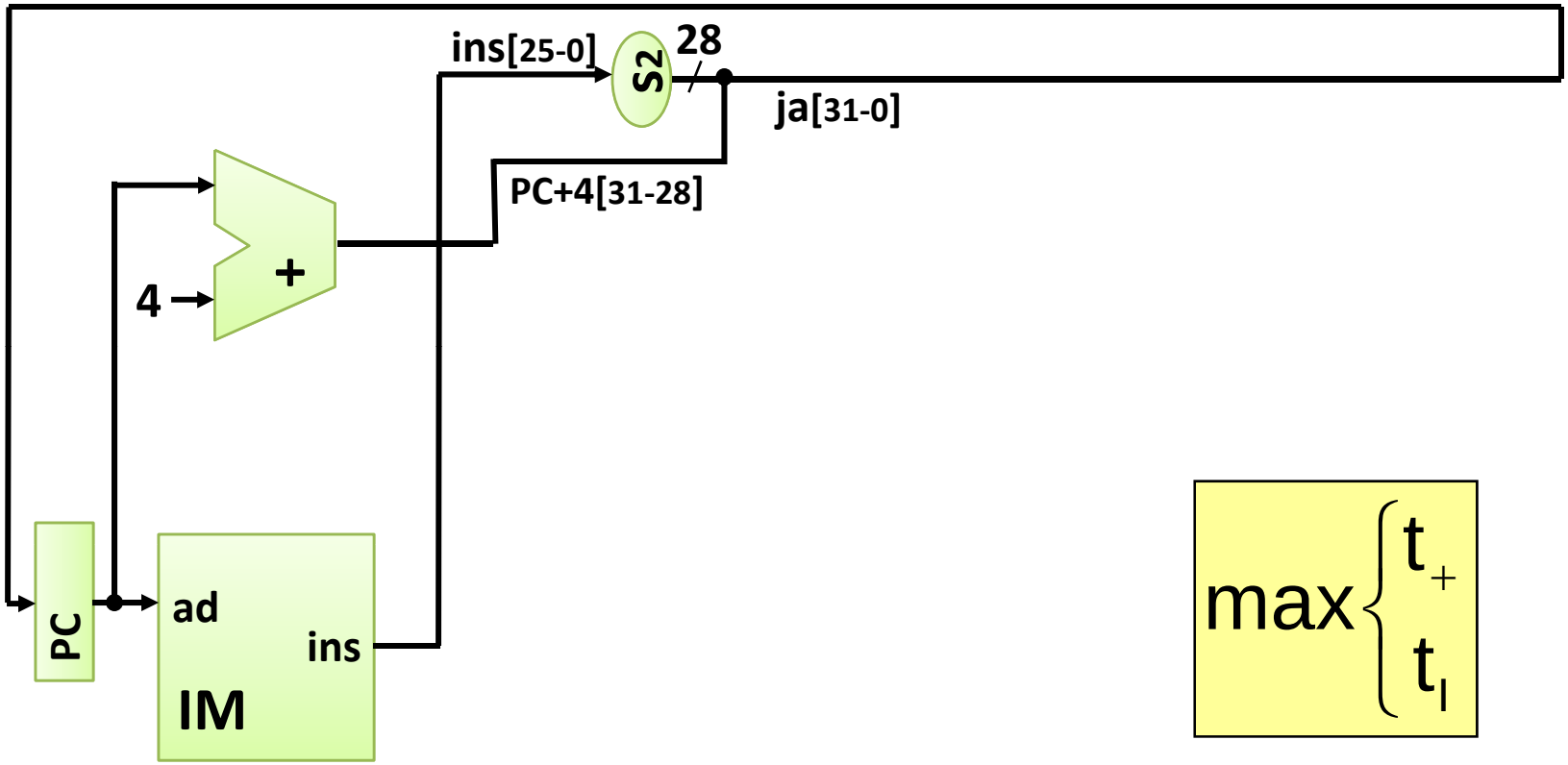
$$\max\begin{cases} t_{+} \\ t_I + t_R + t_A + t_M + t_R \end{cases}$$

Delay for {beq}



$$\max\begin{cases} t_+ + t_+ \\ t_I + t_+ \\ t_I + t_R + t_A \end{cases}$$

Delay for { j}



$$\max\begin{cases} t_{+} \\ t_{I} \end{cases}$$

Overall clock period

$$\max \begin{cases} R : t_+, t_I + t_R + t_A + t_R \\ SW : t_+, t_I + t_R + t_A + t_M \\ LW : t_+, t_I + t_R + t_A + t_M + t_R \\ Beq : t_+ + t_+, t_I + t_+, t_I + t_R + t_A \\ J : t_+, t_I \end{cases}$$

or

$$\max \begin{cases} t_I + t_R + t_A + t_M + t_R \\ t_+ + t_+ \\ t_I + t_+ \end{cases}$$

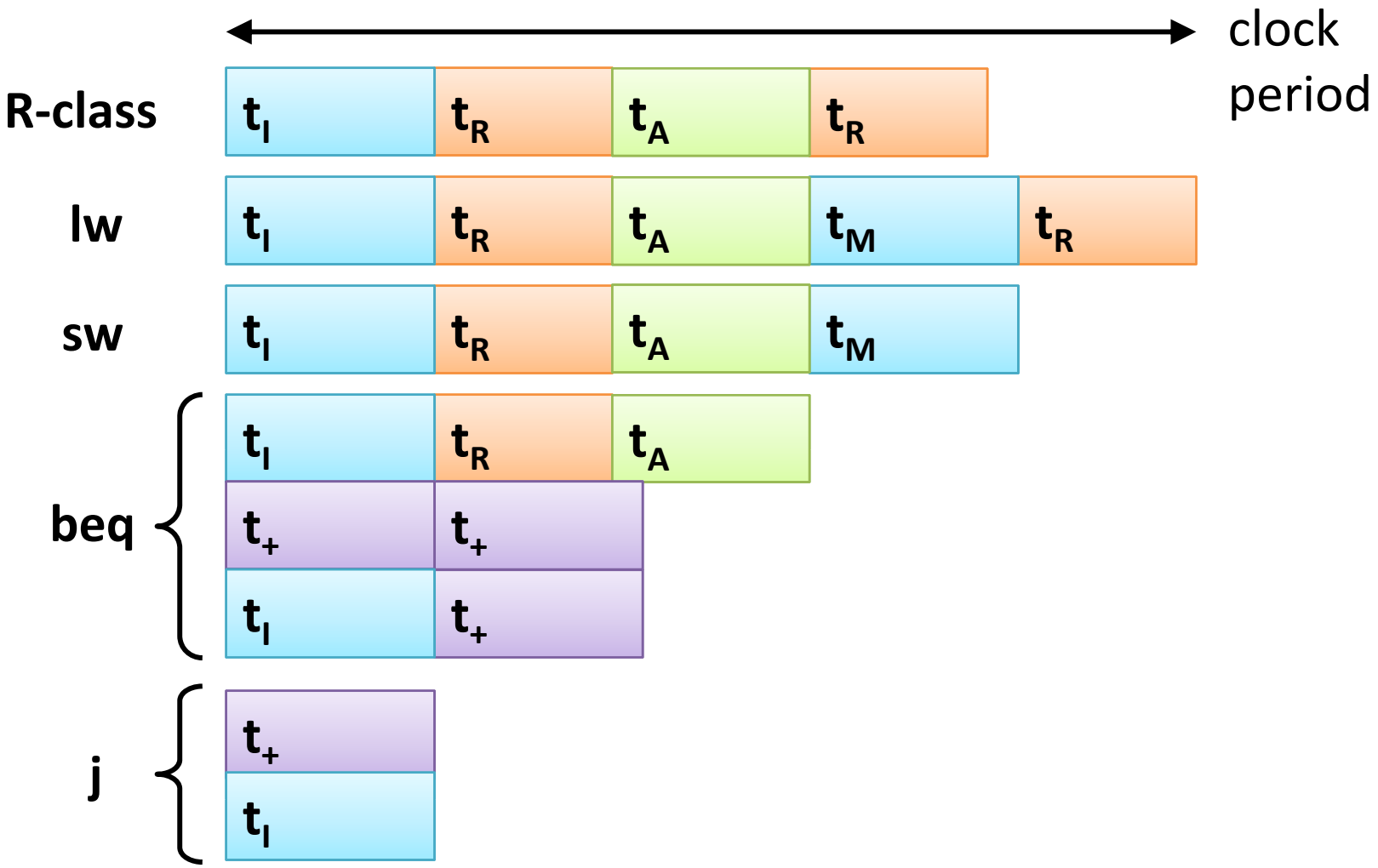
Analyzing performance

Components	Delay	Example
Register	0	**0ns**
Adder	t_+	**4ns**
ALU	t_A	**5ns**
MUX	0	**0ns**
RF	t_R	**3ns**
Instruction Memory	t_I	**6ns**
Data Memory	t_m	**6ns**
Bit manipulation	0	**0ns**

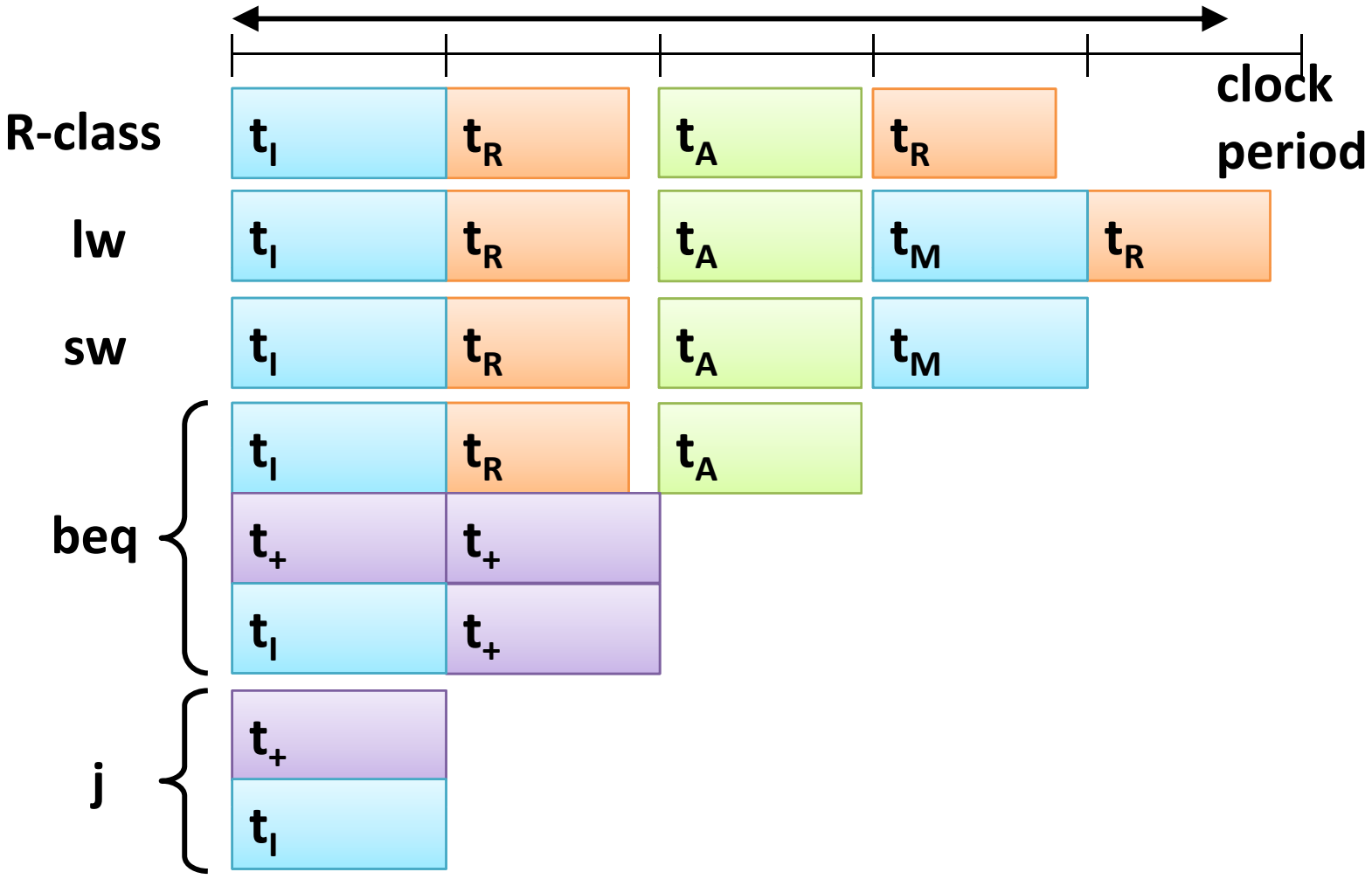
INS	Delay
R	17ns
SW	20ns
LW	23ns
Beq	14ns
J	6ns

Multi-cycle Data Path

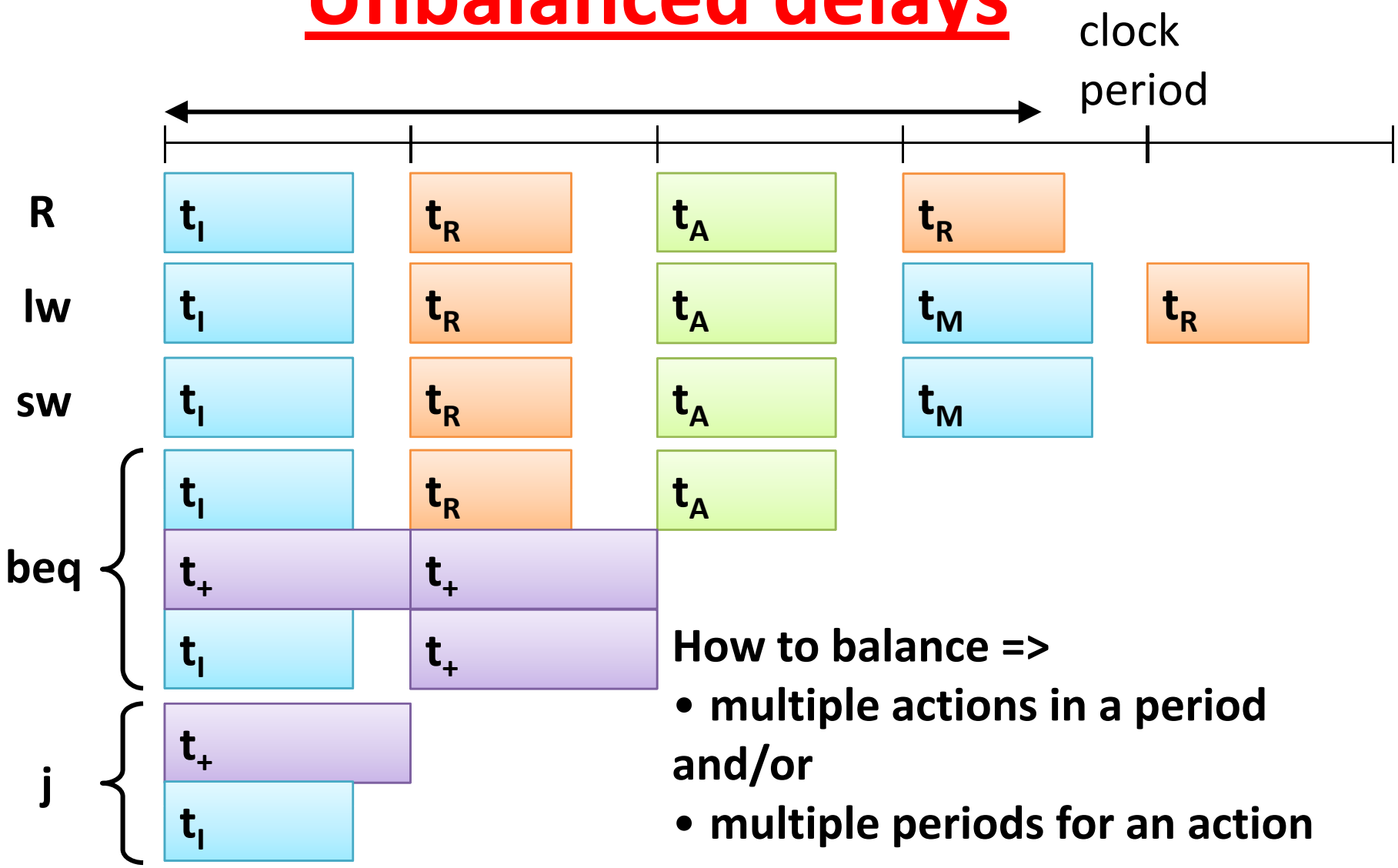
1. Clock period in single cycle design



1. Clock period in multi-cycle design



Unbalanced delays



How to balance =>

- **multiple actions in a period**

and/or

- **multiple periods for an action**

Improving resource utilization

- Can we eliminate two adders?
- How to share (or reuse) a resource (say ALU) in different clock cycles?

- Store results in registers.
- Of course, more multiplexing may be required!
- Resources in this design: RF, ALU, MEM.

Thanks