

Архитектура Современных Процессоров

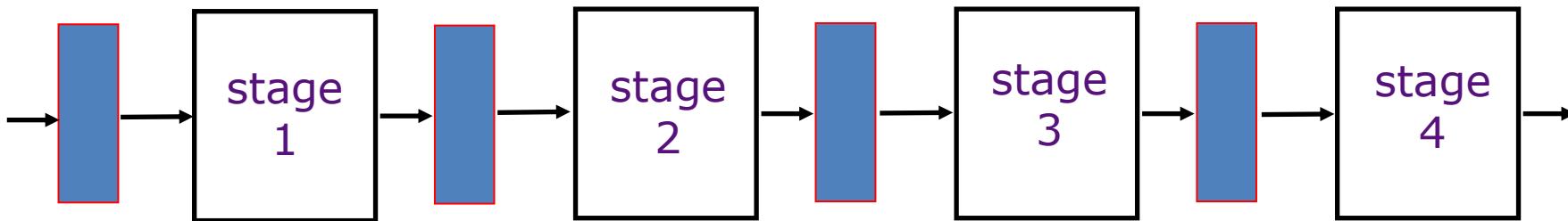
Modern Processor Architecture

Лекция 6: Конфликты и прерывания в конвейере
скалярного процессора

Кожин Алексей Сергеевич

E-mail: aleksej.kozhin@gmail.com

Свойства идеального конвейера



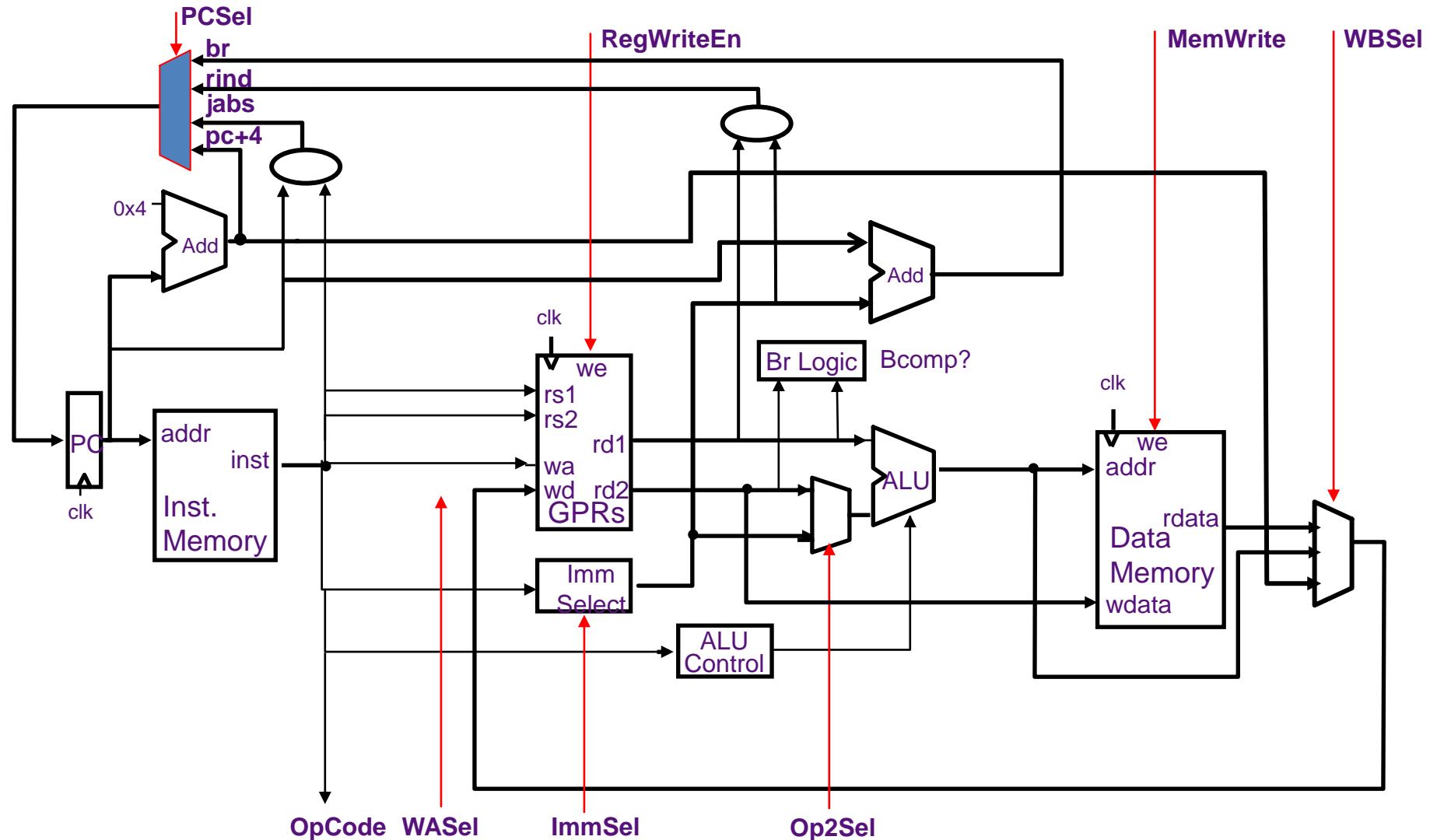
- Все команды проходят через одни и те же стадии
- Между двумя стадиями нет общего оборудования, всё передаётся через регистровые станции
- Задержки и накладные расходы на всех стадиях конвейера и время, необходимое на работу всех стадий конвейера одинаковые
- Порядок следования запросов по различным стадиям конвейера не имеет значения. То есть, команды, находящиеся на других стадиях конвейера, не влияют на выполнение данной стадии конвейера

... но в жизни не все так хорошо

Конвейеризованный RISC-V

- Чтобы конвейеризовать RISC-V:
- Сначала нужно построить RISC-V без конвейеризации с $CPI = 1$
- Затем добавить конвейерные регистры, чтобы уменьшить длительность такта, сохраняя при этом $CPI = 1$

Неконвейеризованный тракт данных для RISC-V



Hardwired Control Table

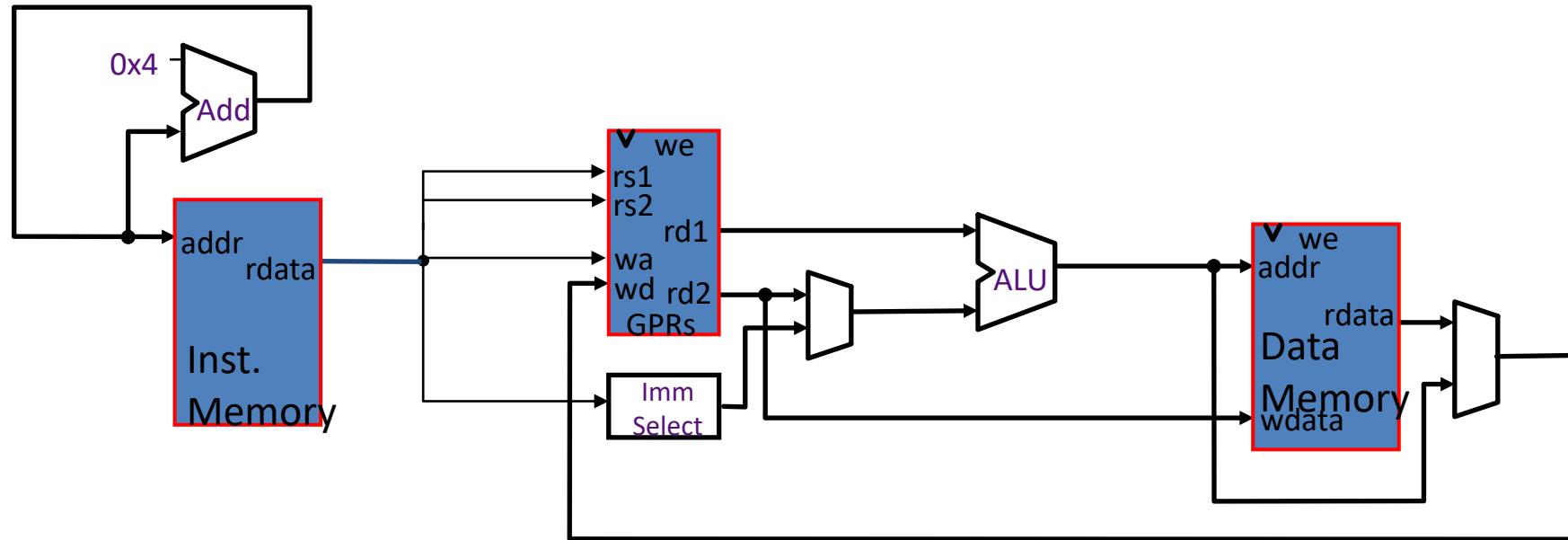
Opcode	ImmSel	Op2Sel	FuncSel	MemWr	RFWen	WBSel	WASel	PCSel
ALU	*	Reg	Func	no	yes	ALU	rd	pc+4
ALUi	IType ₁₂	Imm	Op	no	yes	ALU	rd	pc+4
LW	IType ₁₂	Imm	+	no	yes	Mem	rd	pc+4
SW	BsType ₁₂	Imm	+	yes	no	*	*	pc+4
BEQ _{true}	BrType ₁₂	*	*	no	no	*	*	br
BEQ _{false}	BrType ₁₂	*	*	no	no	*	*	pc+4
JAL	*	*	*	no	yes	PC	rd	jabs
JALR	*	*	*	no	yes	PC	rd	rind

Op2Sel= Reg / Imm

WBSel = ALU / Mem / PC

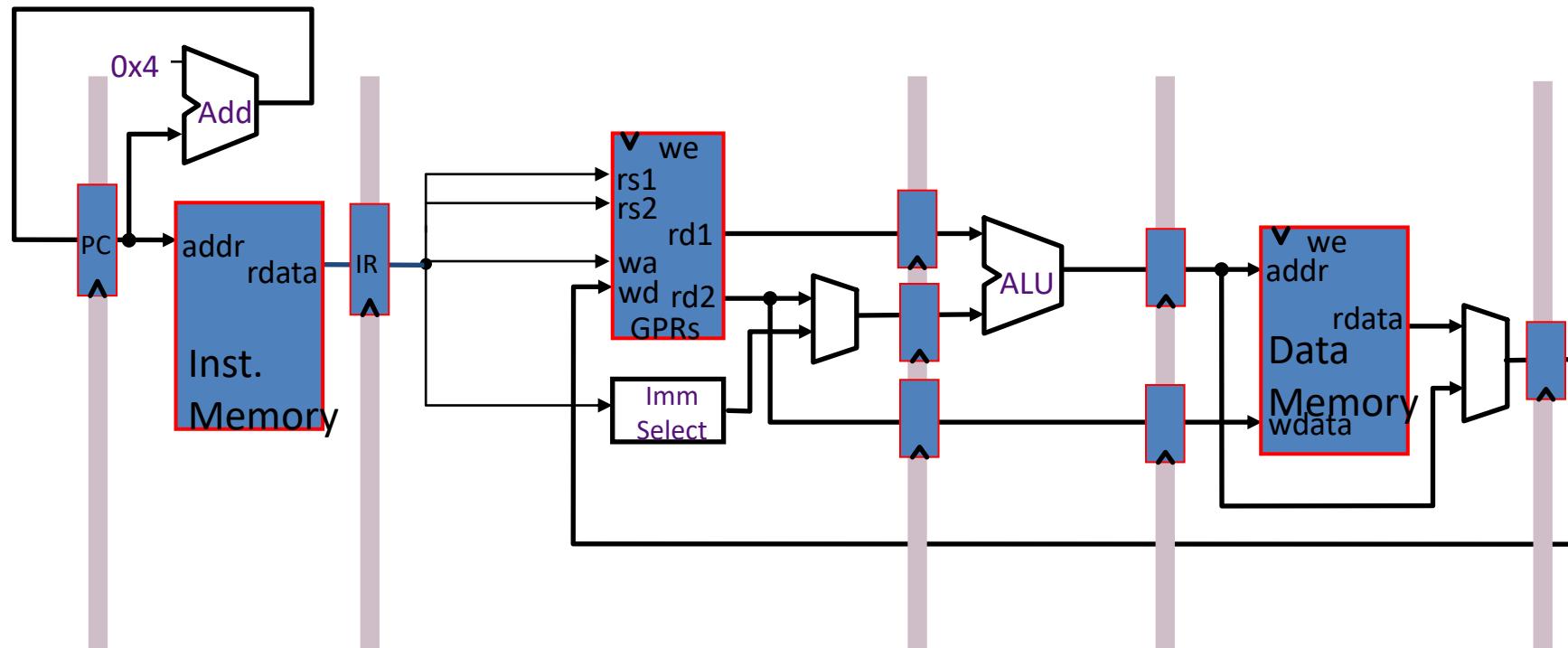
PCSel = pc+4 / br / rind / jabs

Конвейеризованный тракт данных



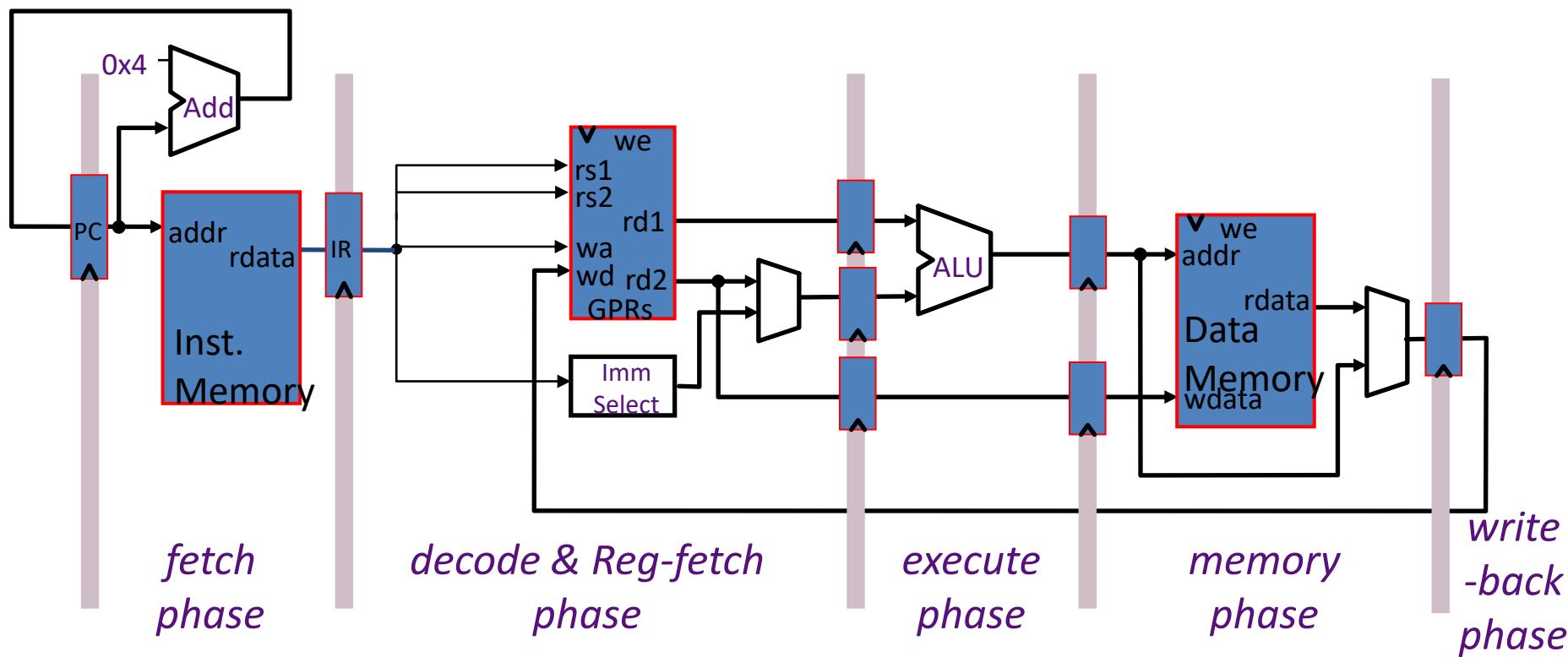
Но CPI возрастет, если инструкции не будут конвейеризованы

Конвейеризованный тракт данных



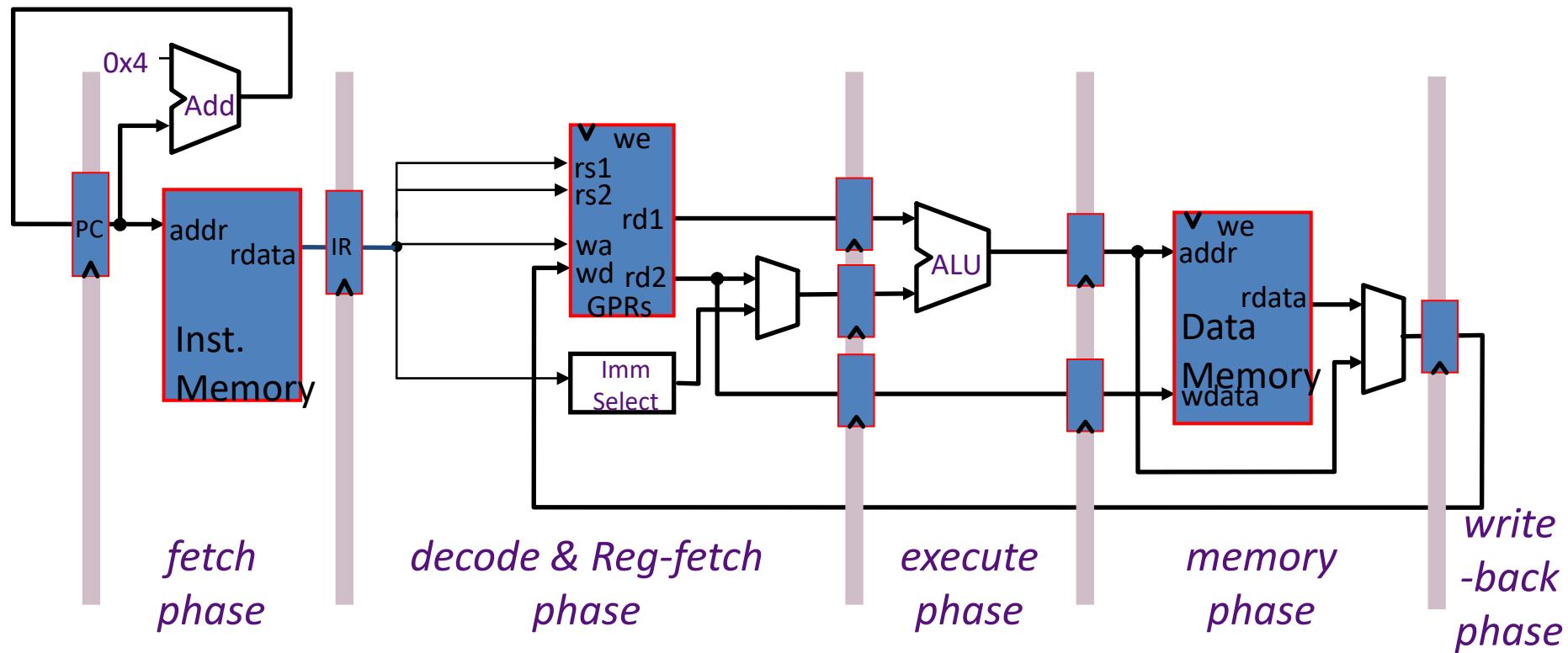
Но CPI возрастет, если инструкции не будут конвейеризованы

Конвейеризованный тракт данных



Но CPI возрастет, если инструкции не будут конвейеризованы

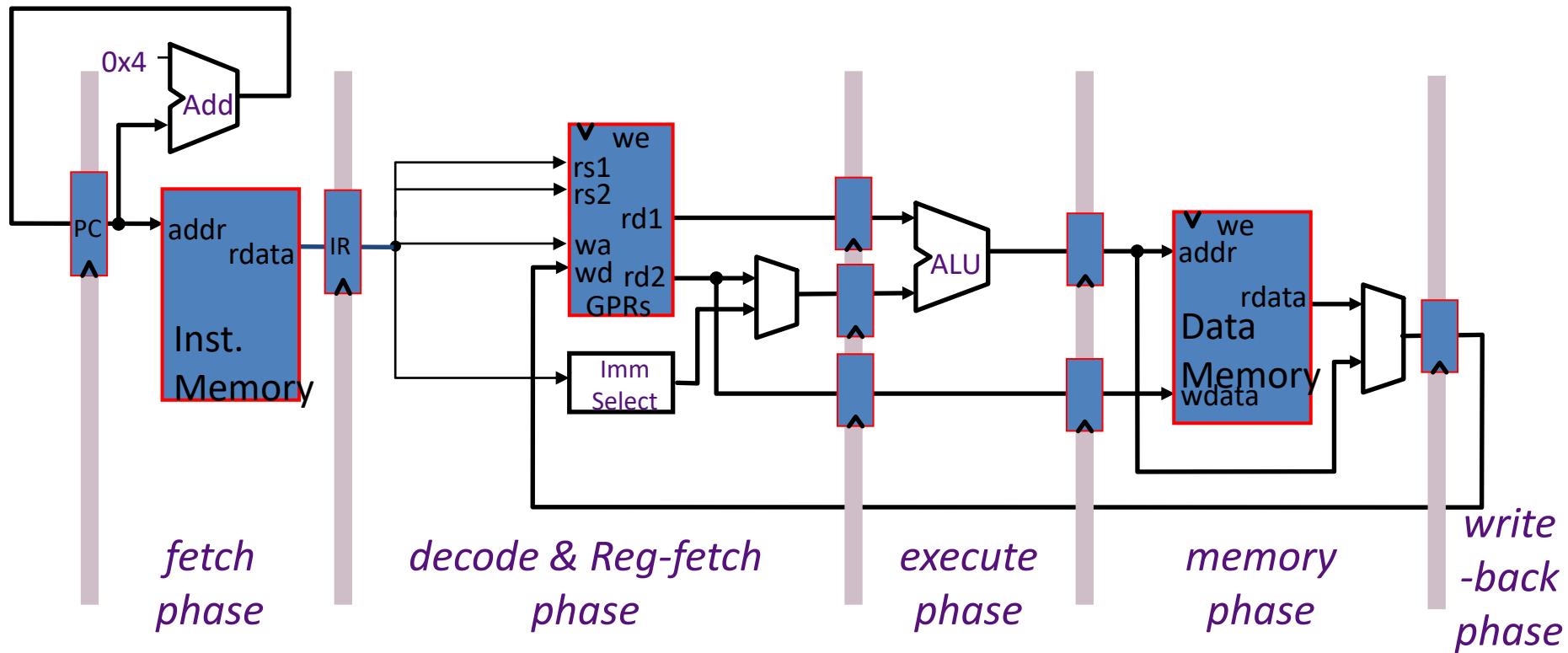
Конвейеризованный тракт данных



Длительность такта может быть снижена путем разделения исполнения инструкции на несколько тактов

Но CPI возрастет, если инструкции не будут конвейеризованы

Конвейеризованный тракт данных



Длительность такта может быть снижена путем разделения исполнения инструкции на несколько тактов

$$t_C > \max \{t_{IM}, t_{RF}, t_{ALU}, t_{DM}, t_{RW}\} \text{ (скорее всего, } = t_{DM})$$

Но CPI возрастет, если инструкции не будут конвейеризованы

“Iron Law” производительности

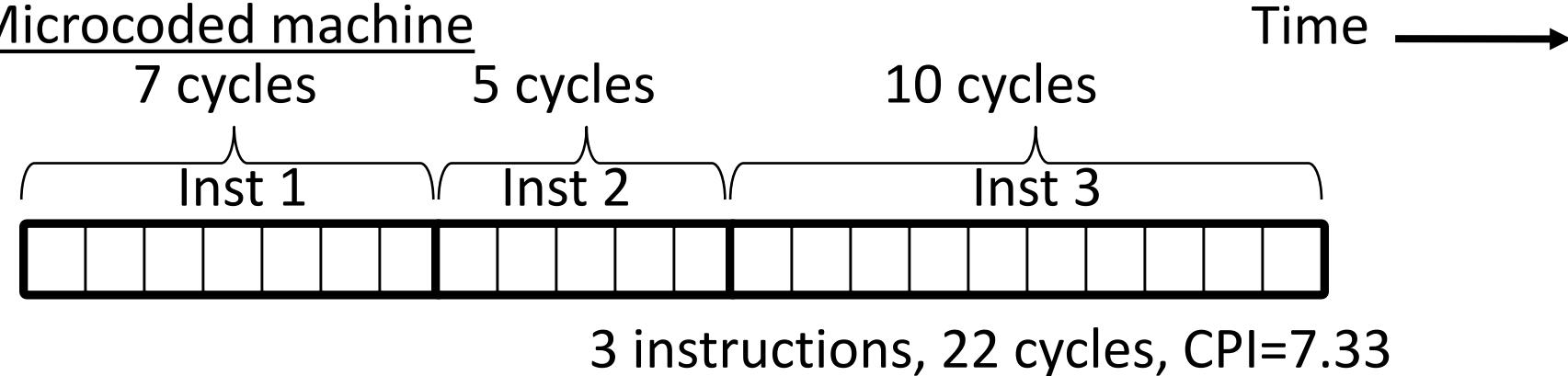
$$\frac{\text{Time}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} * \frac{\text{Cycles}}{\text{Instruction}} * \frac{\text{Time}}{\text{Cycle}}$$

- Количество инструкций, из которых состоит программа (зависит от программы, компилятора, и архитектуры набора команд ISA)
- Количество тактов процессорного времени, необходимых на выполнение одной инструкции CPI (зависит от ISA и микроархитектуры)
- Тактовая частоты процессора (микроархитектура и технология изготовления)

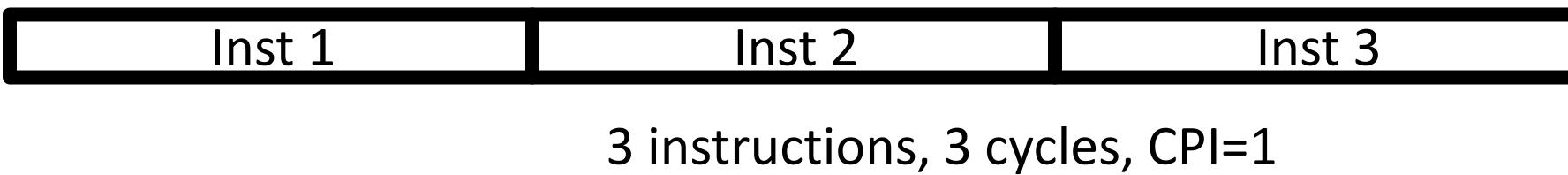
Microarchitecture	CPI	cycle time
Microcoded	>1	short
Single-cycle unpipelined	1	long
Pipelined	1	short

Примеры CPI

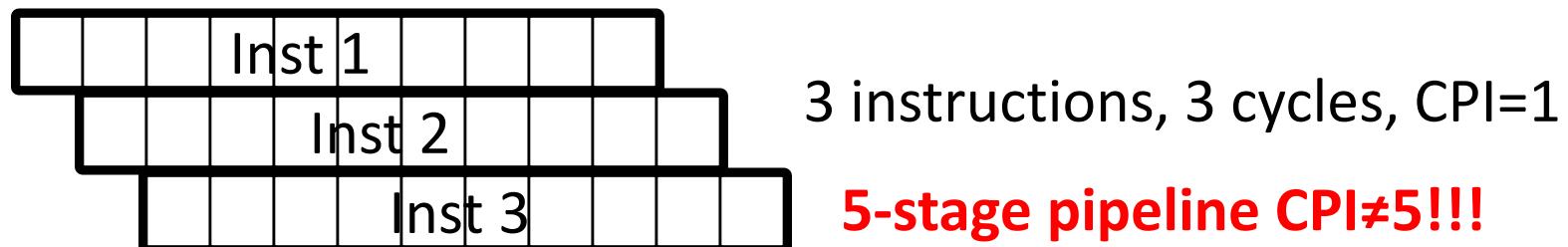
Microcoded machine



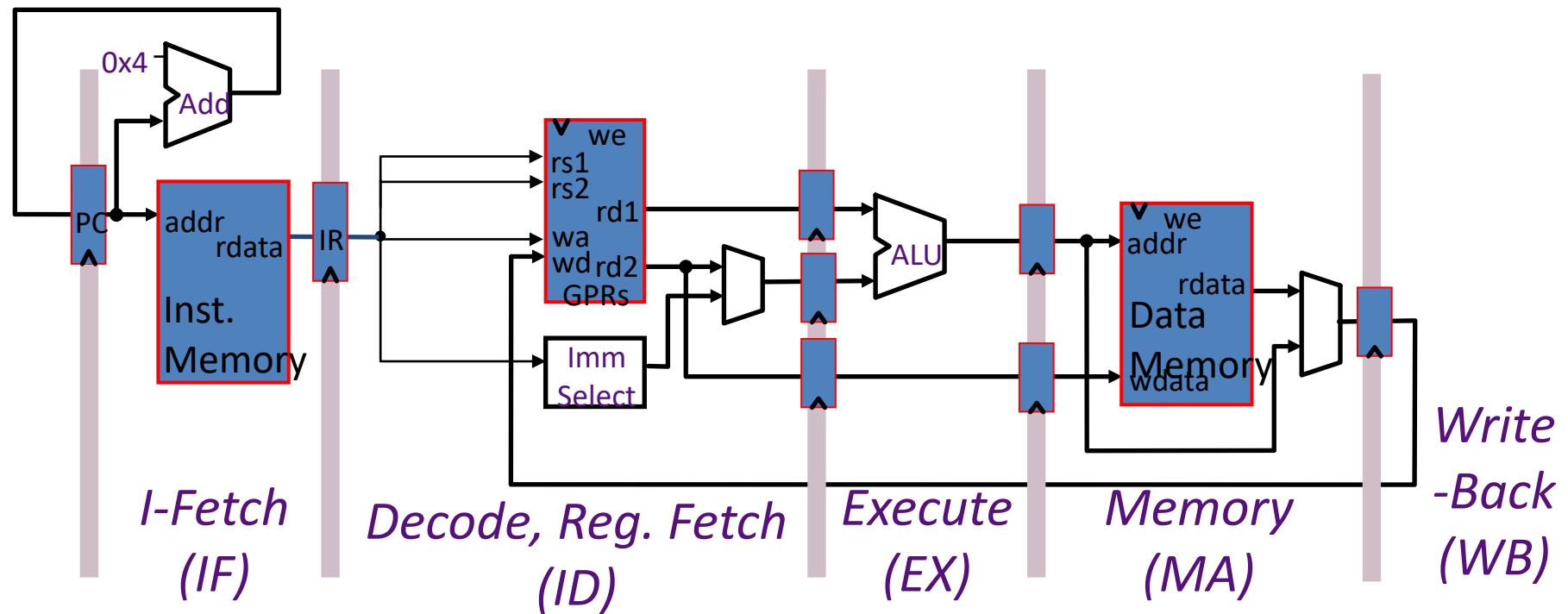
Unpipelined machine



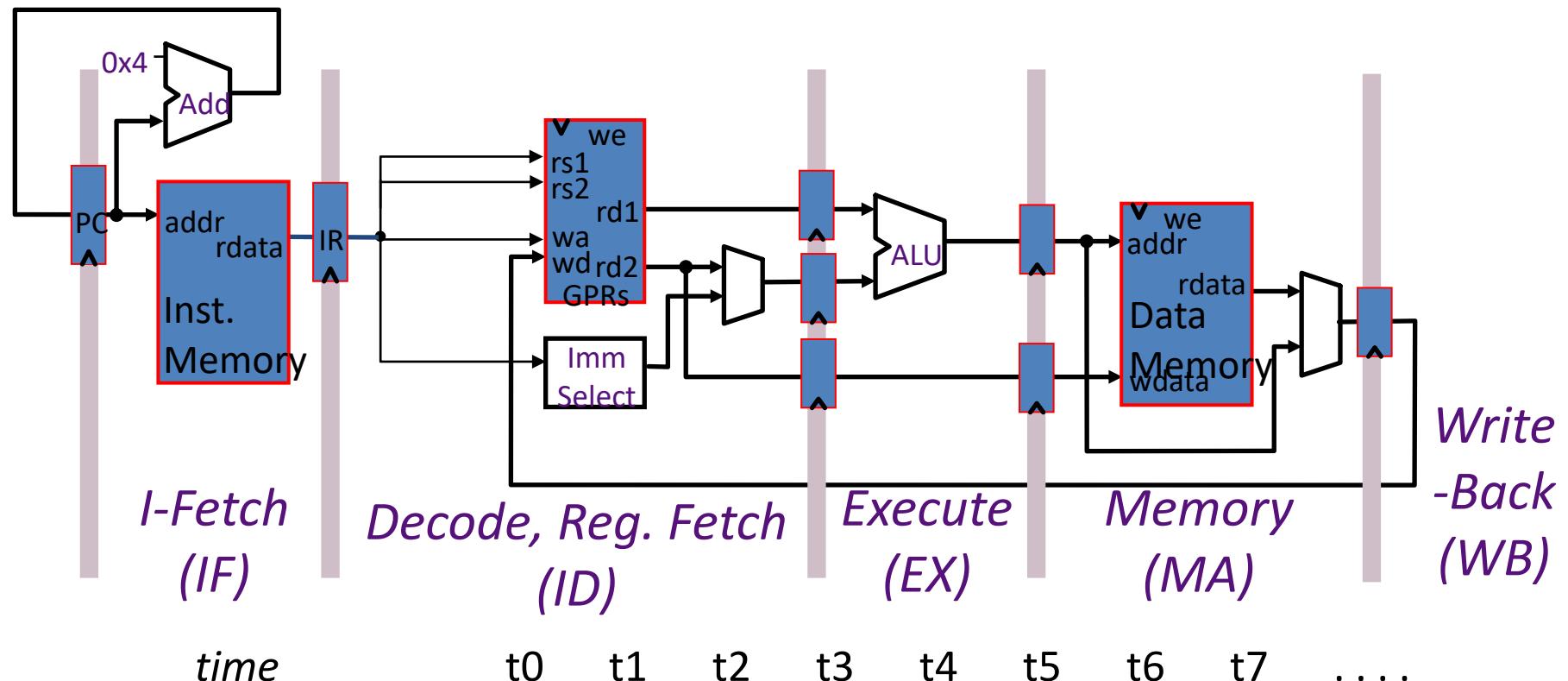
Pipelined machine



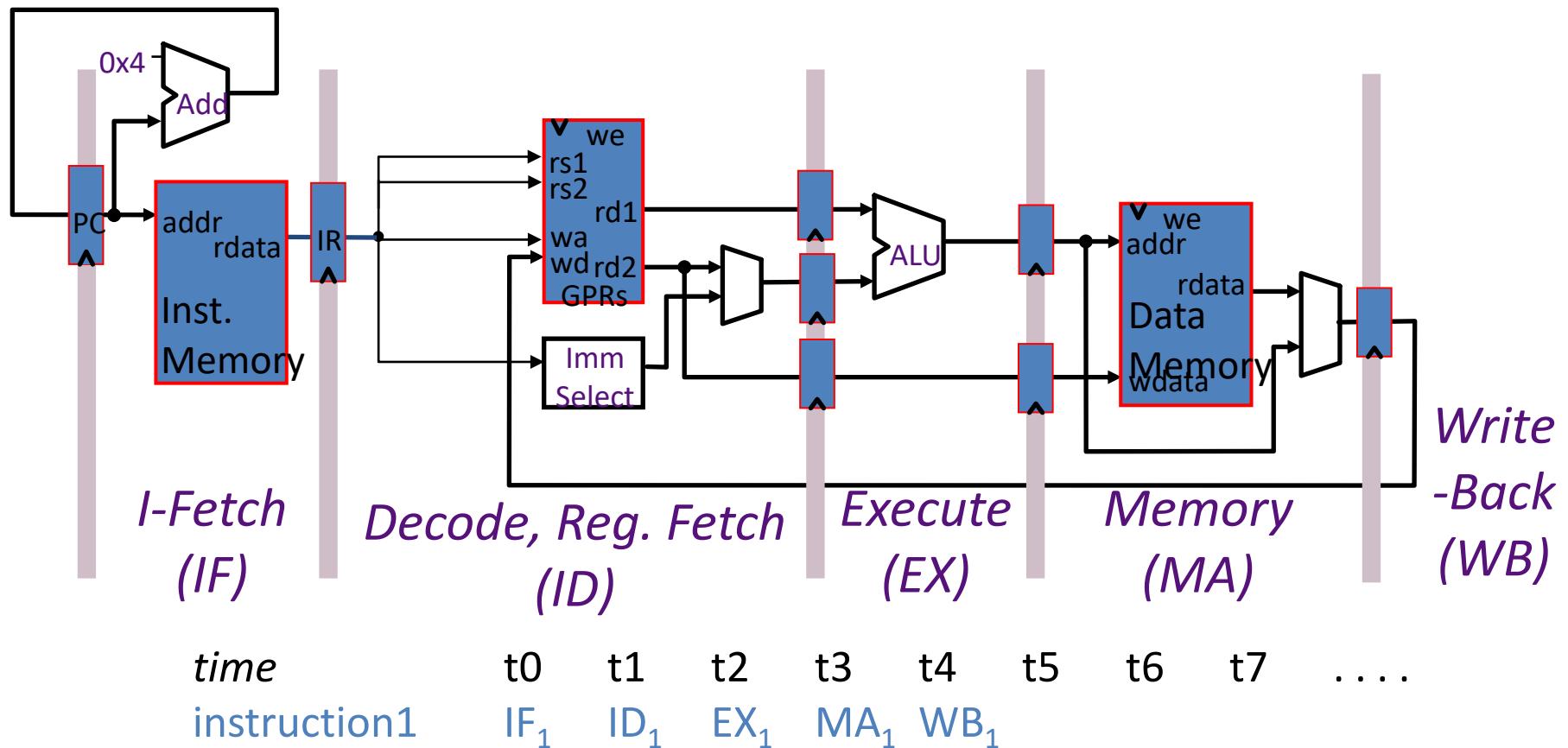
Пятистадийное конвейерное исполнение команд



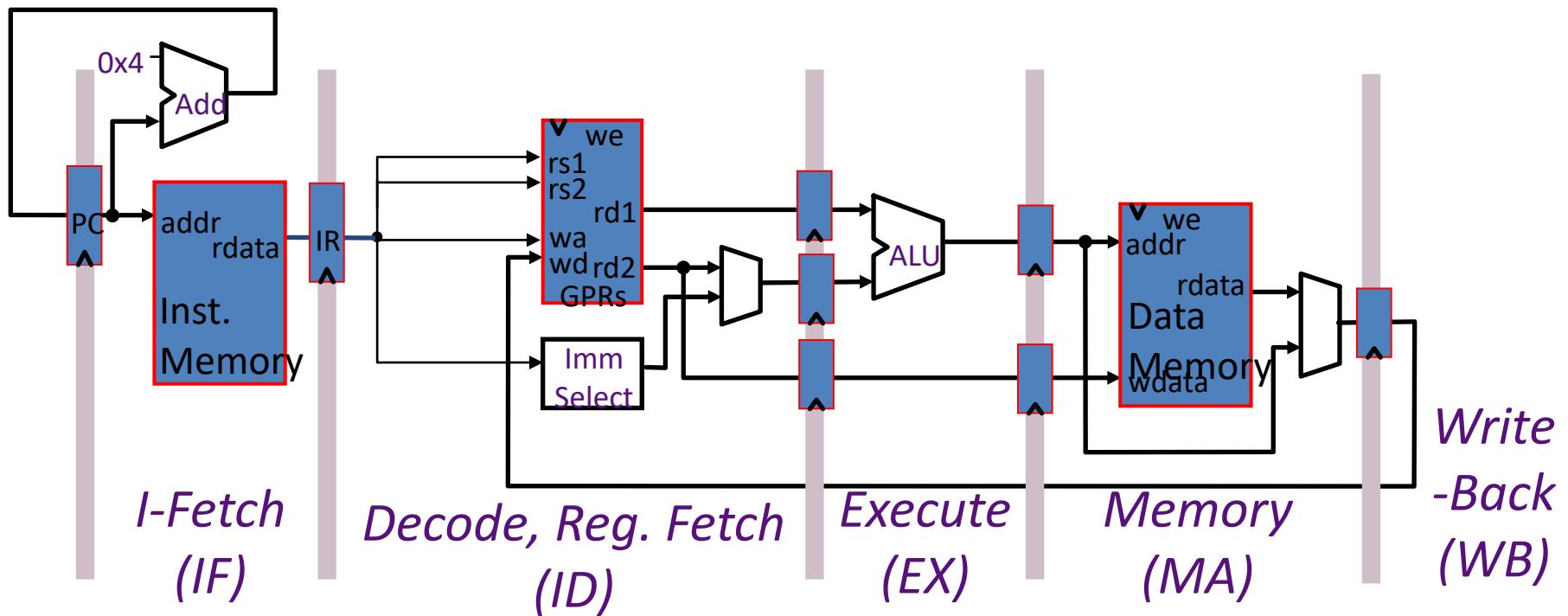
Пятистадийное конвейерное исполнение команд



Пятистадийное конвейерное исполнение команд

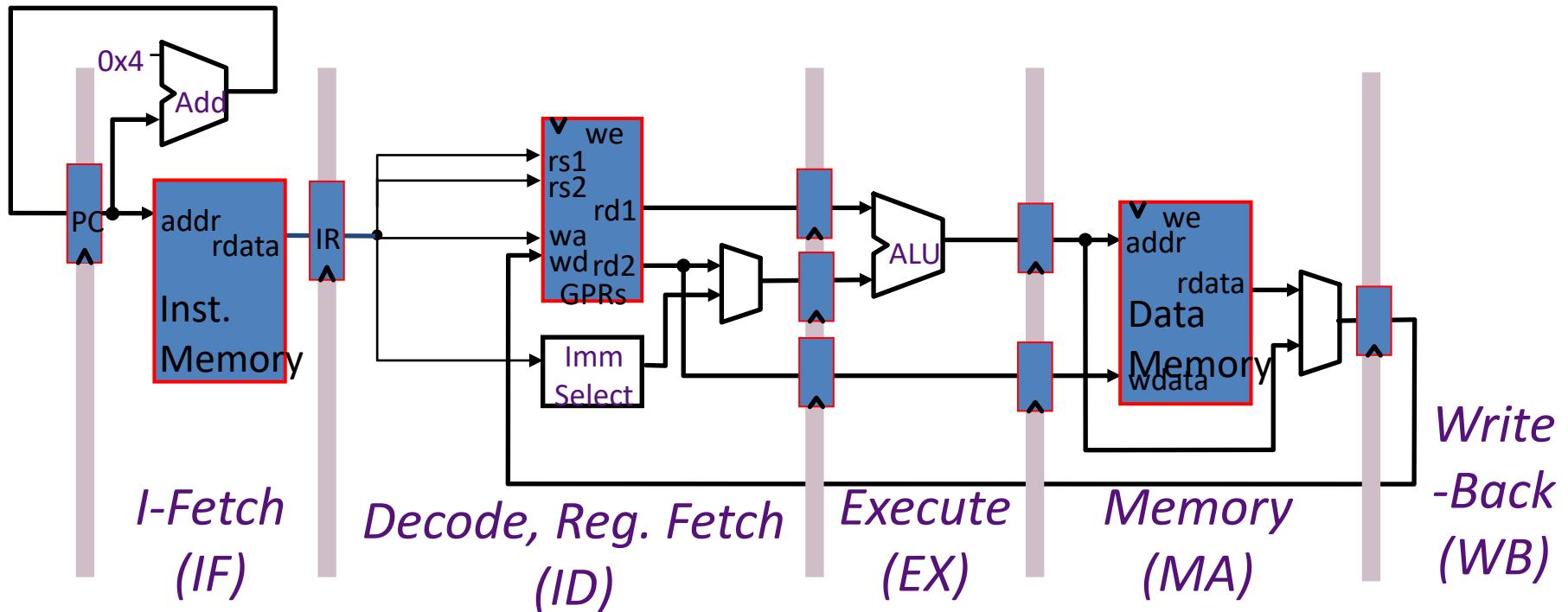


Пятистадийное конвейерное исполнение команд



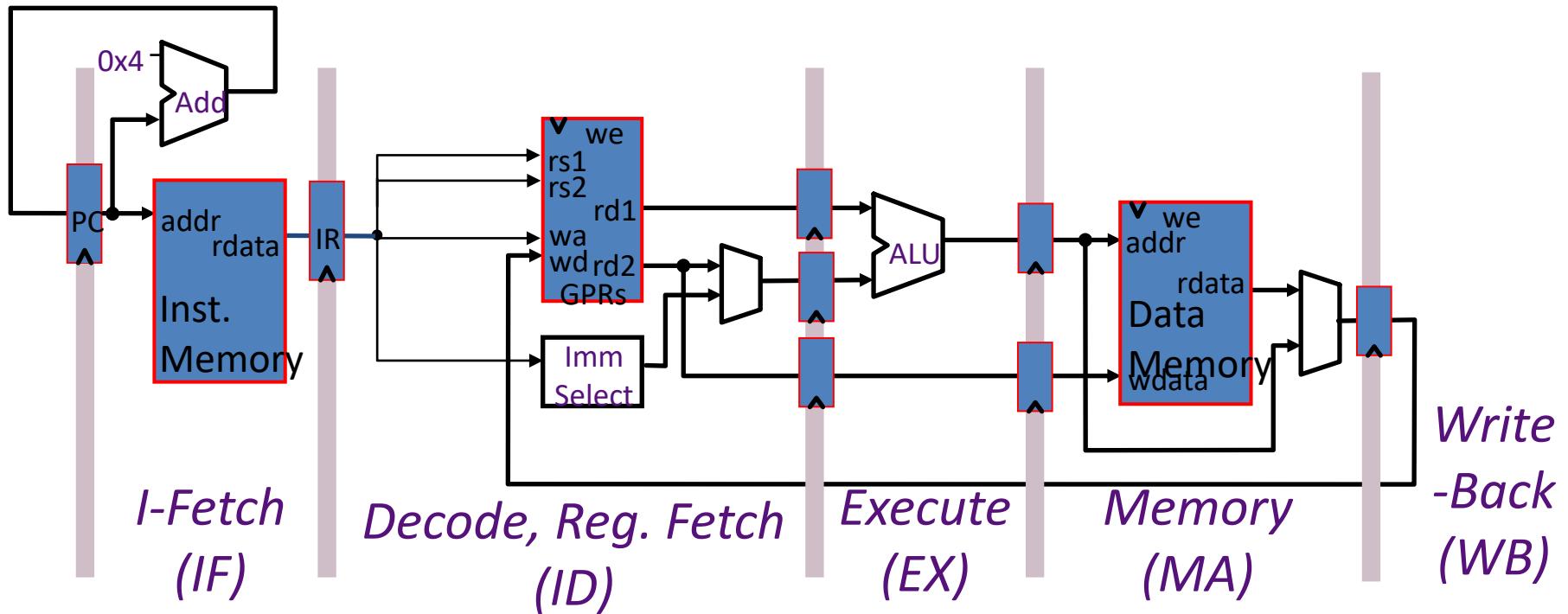
<i>time</i>	t0	t1	t2	t3	t4	t5	t6	t7
instruction1	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
instruction2		IF ₂	ID ₂	EX ₂	MA ₂	WB ₂			

Пятистадийное конвейерное исполнение команд



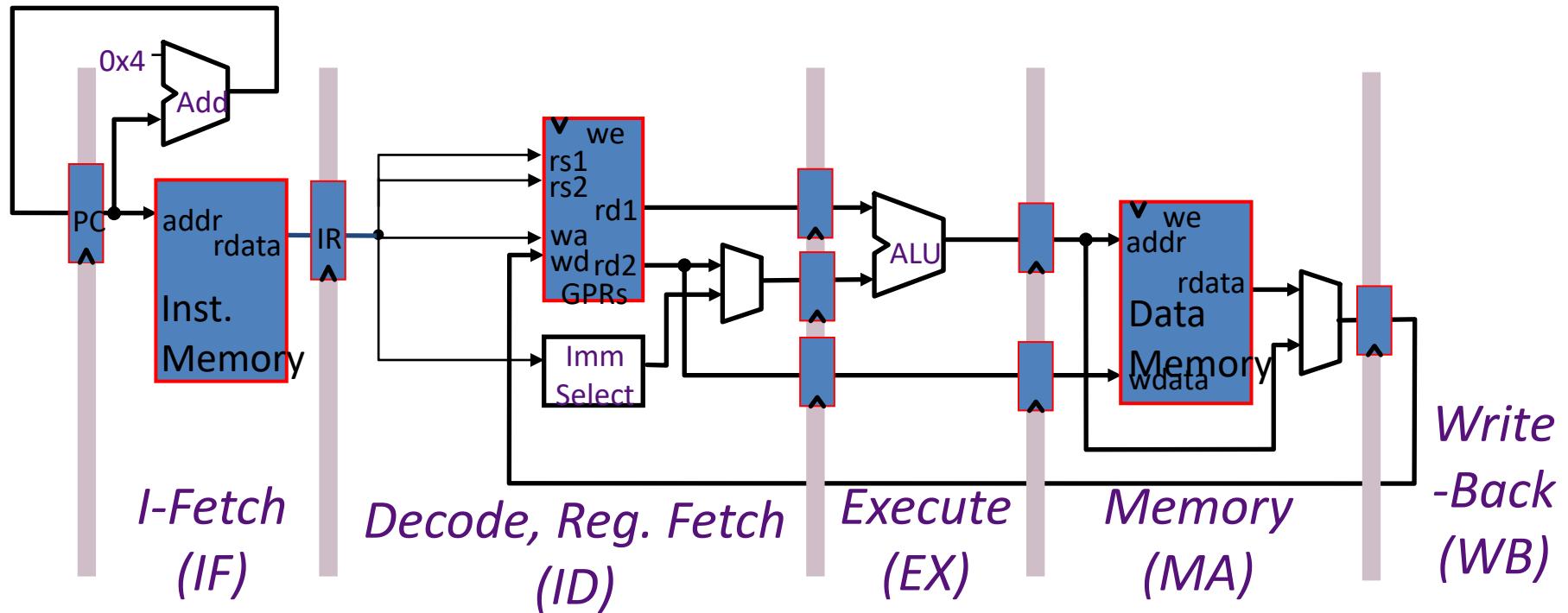
<i>time</i>	t0	t1	t2	t3	t4	t5	t6	t7
instruction1	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
instruction2		IF ₂	ID ₂	EX ₂	MA ₂	WB ₂			
instruction3		IF ₃	ID ₃	EX ₃	MA ₃	WB ₃			

Пятистадийное конвейерное исполнение команд



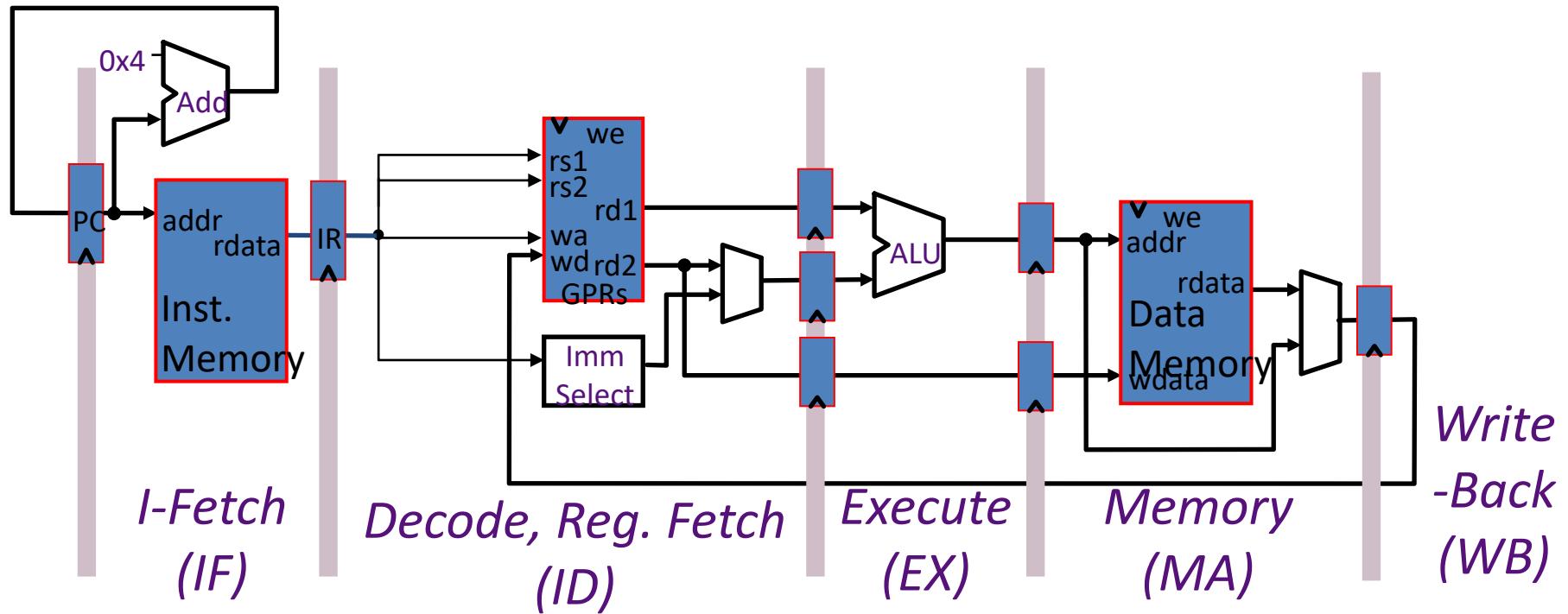
<i>time</i>	t0	t1	t2	t3	t4	t5	t6	t7
instruction1	IF_1	ID_1	EX_1	MA_1	WB_1				
instruction2		ID_2	EX_2	MA_2	WB_2				
instruction3			ID_3	EX_3	MA_3	WB_3			
instruction4			ID_4	EX_4	MA_4	WB_4			

Пятистадийное конвейерное исполнение команд



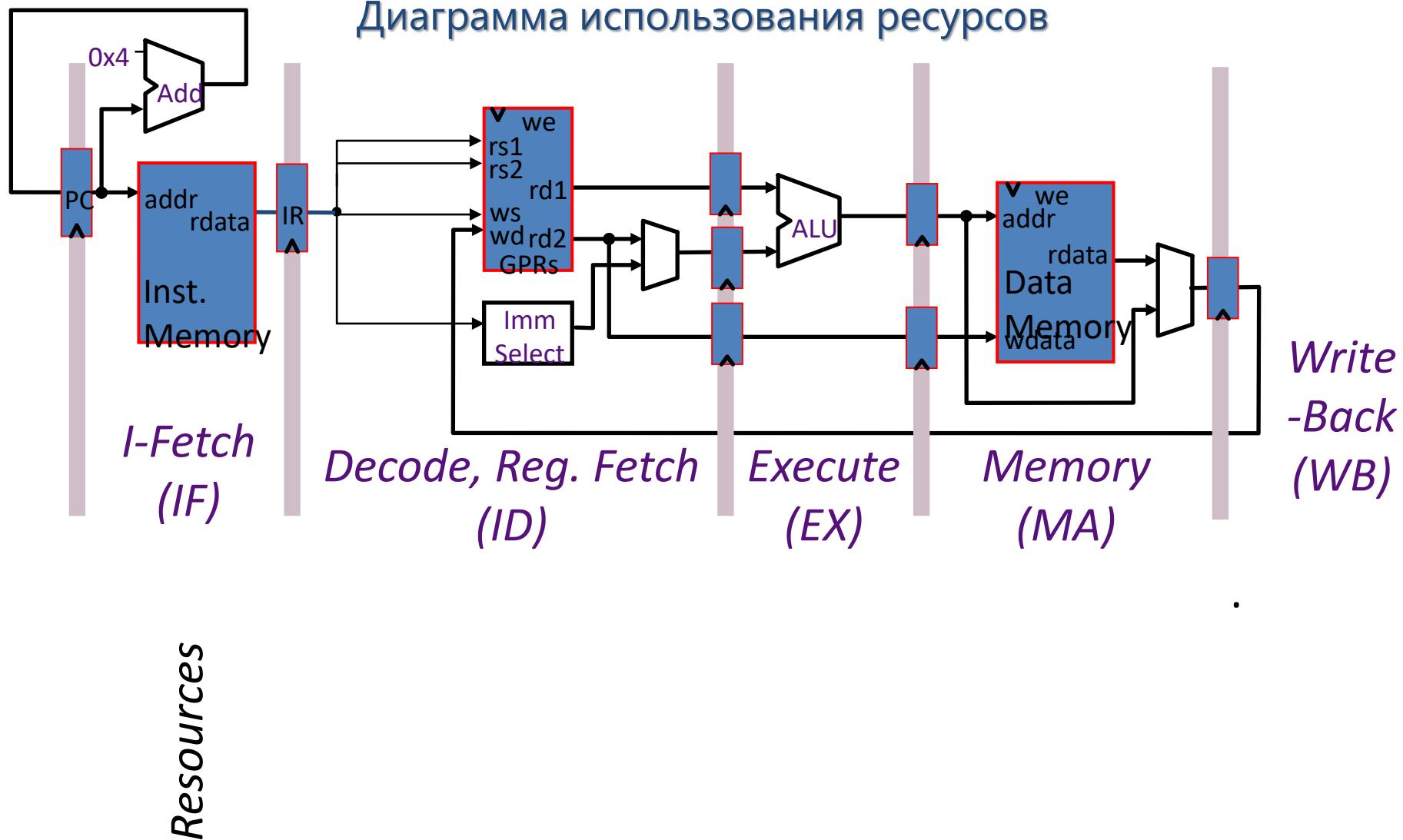
<i>time</i>	t0	t1	t2	t3	t4	t5	t6	t7
instruction1	IF_1	ID_1	EX_1	MA_1	WB_1				
instruction2		ID_2	EX_2	MA_2	WB_2				
instruction3		ID_3	EX_3	MA_3	WB_3				
instruction4		ID_4	EX_4	MA_4	WB_4				
instruction5		ID_5	EX_5	MA_5	WB_5				

Пятистадийное конвейерное исполнение команд

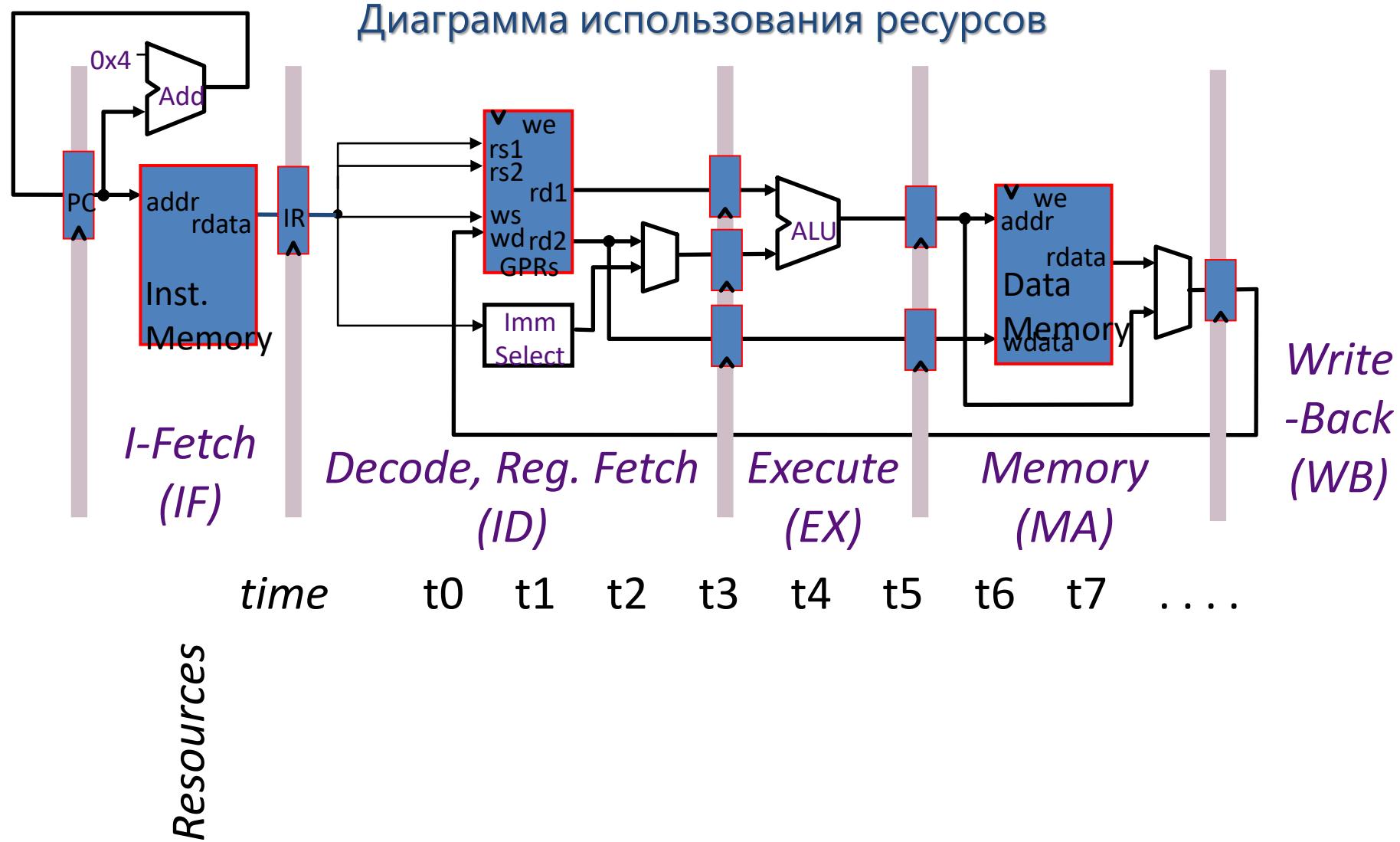


<i>time</i>	t0	t1	t2	t3	t4	t5	t6	t7
instruction1	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
instruction2		IF ₂	ID ₂	EX ₂	MA ₂	WB ₂			
instruction3			IF ₃	ID ₃	EX ₃	MA ₃	WB ₃		
instruction4				IF ₄	ID ₄	EX ₄	MA ₄	WB ₄	
instruction5					IF ₅	ID ₅	EX ₅	MA ₅	WB ₅

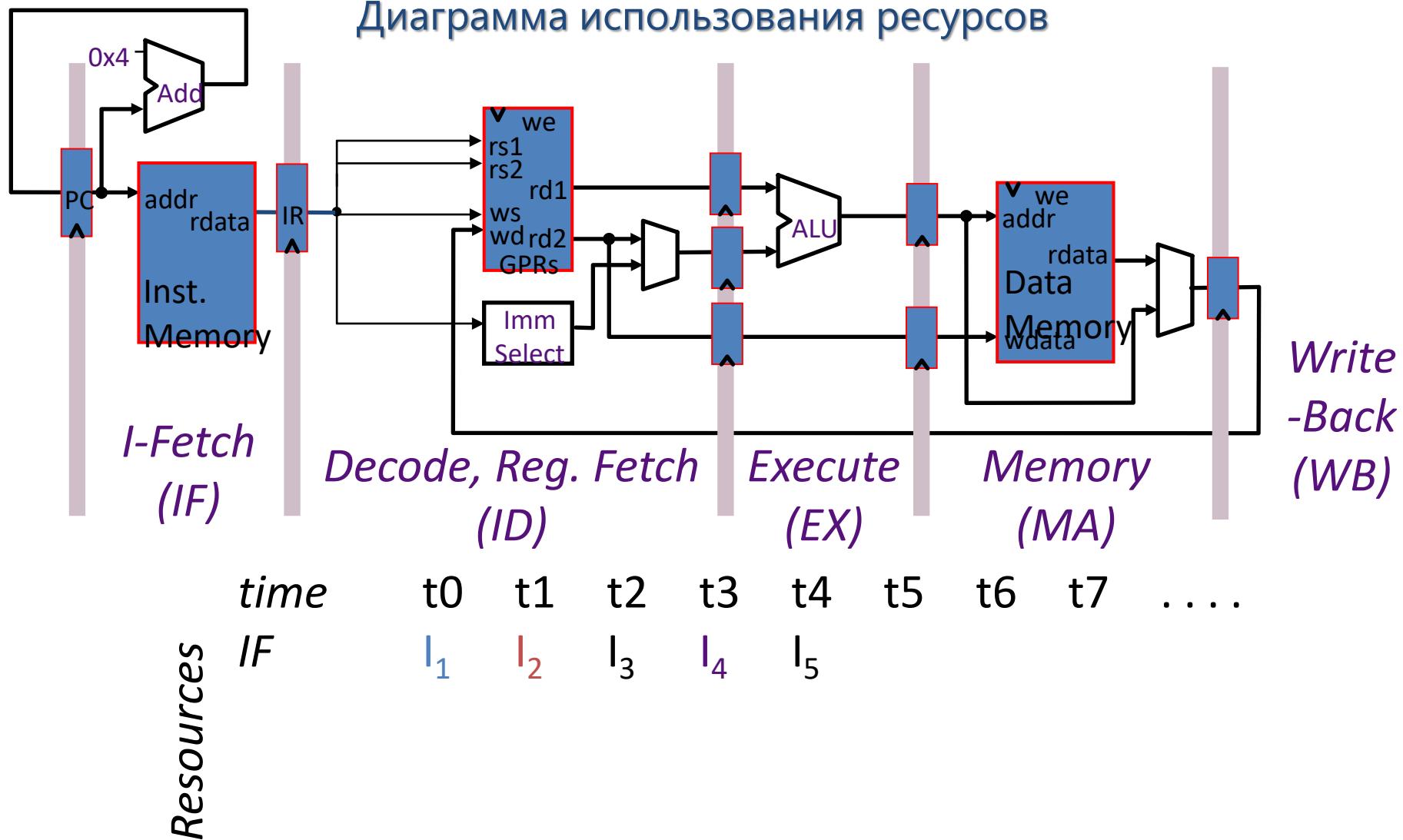
Пятистадийное конвейерное исполнение команд



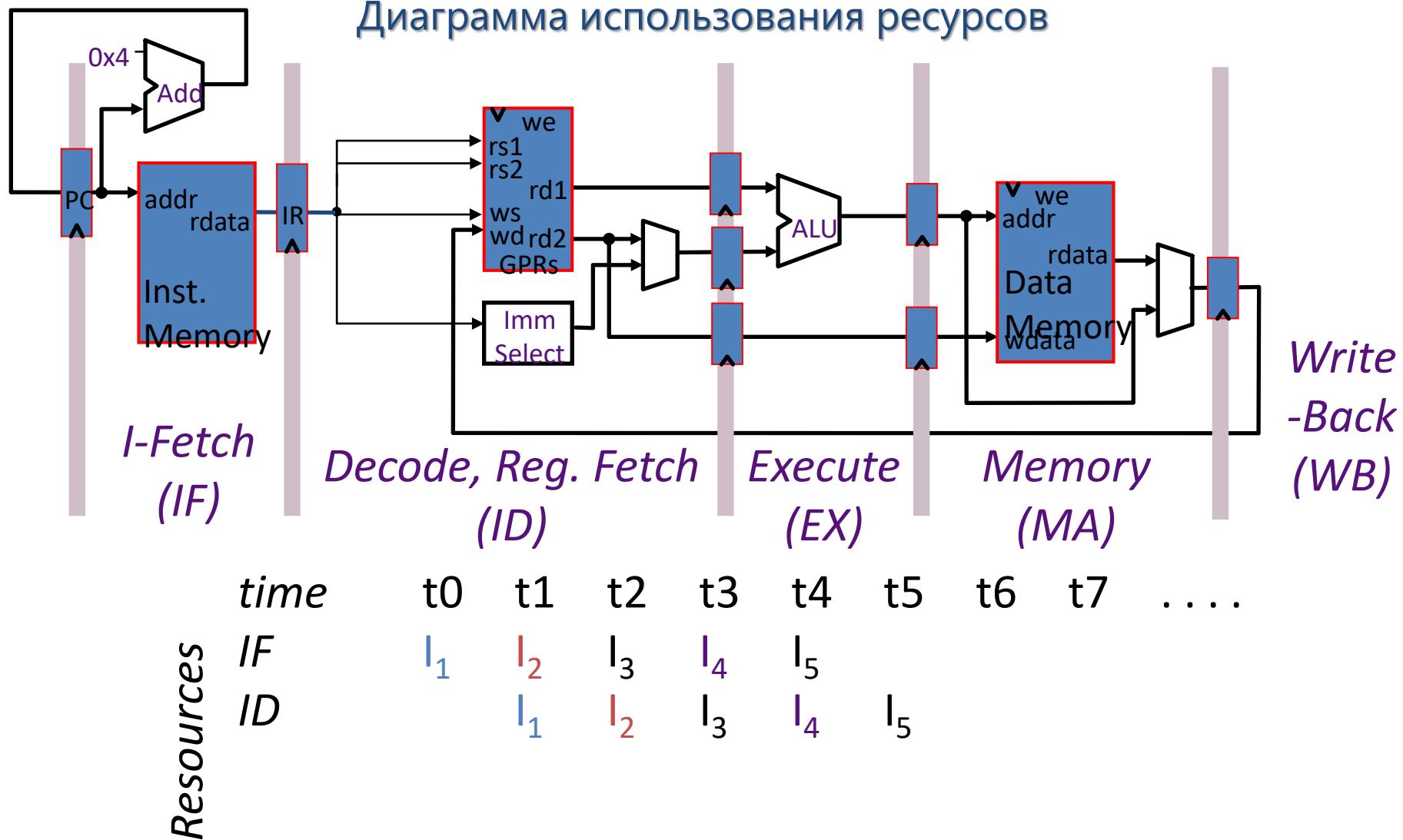
Пятистадийное конвейерное исполнение команд



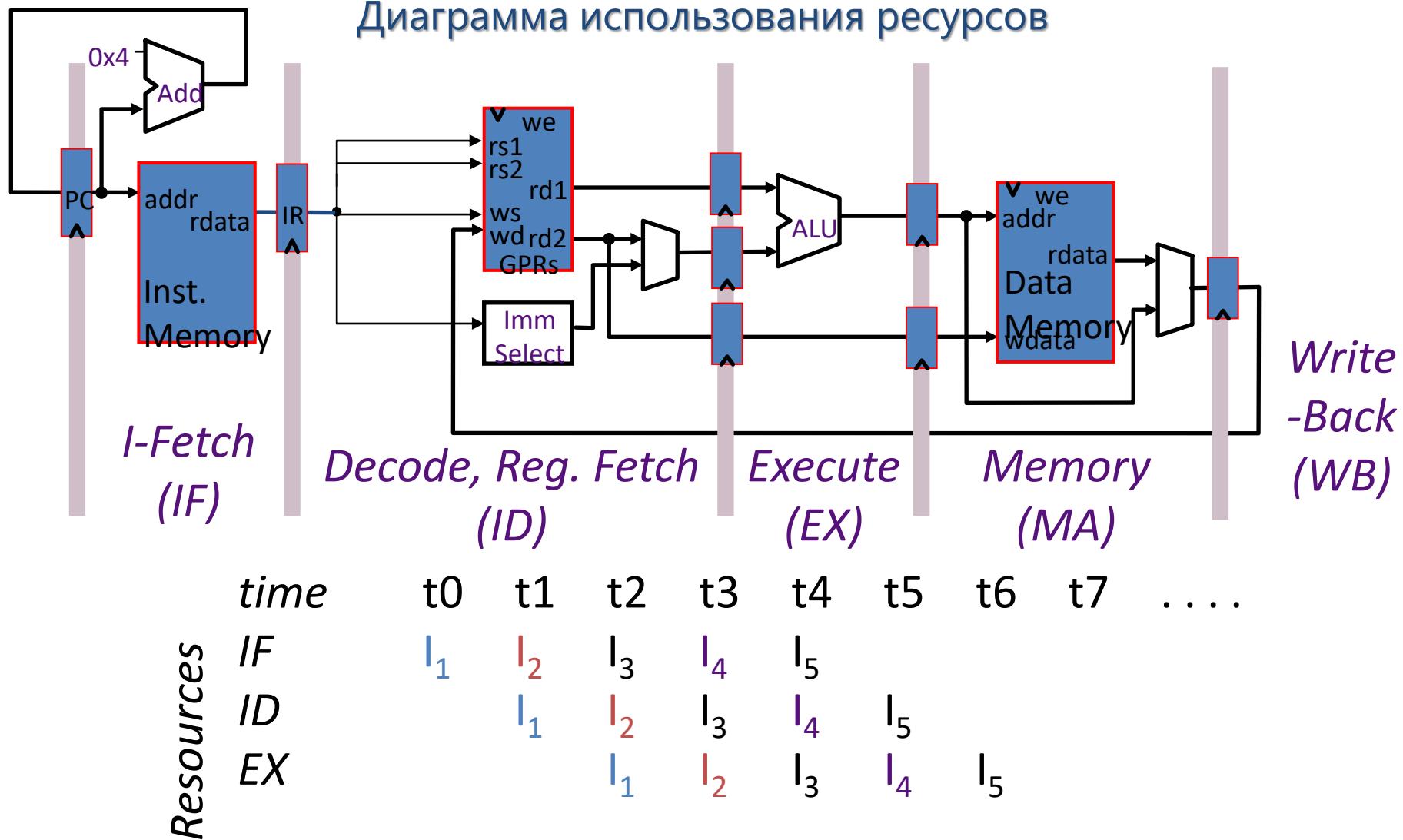
Пятистадийное конвейерное исполнение команд



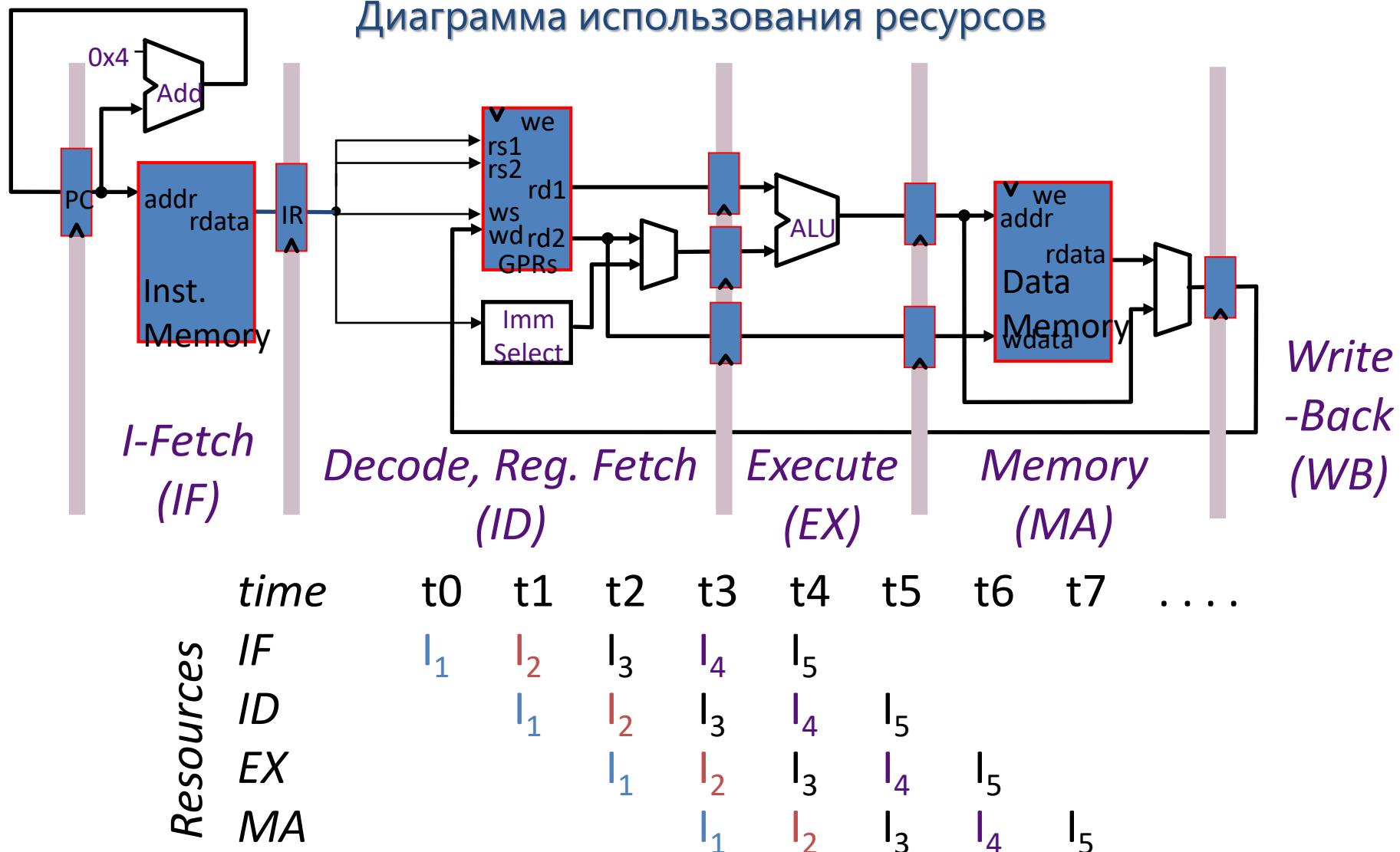
Пятистадийное конвейерное исполнение команд



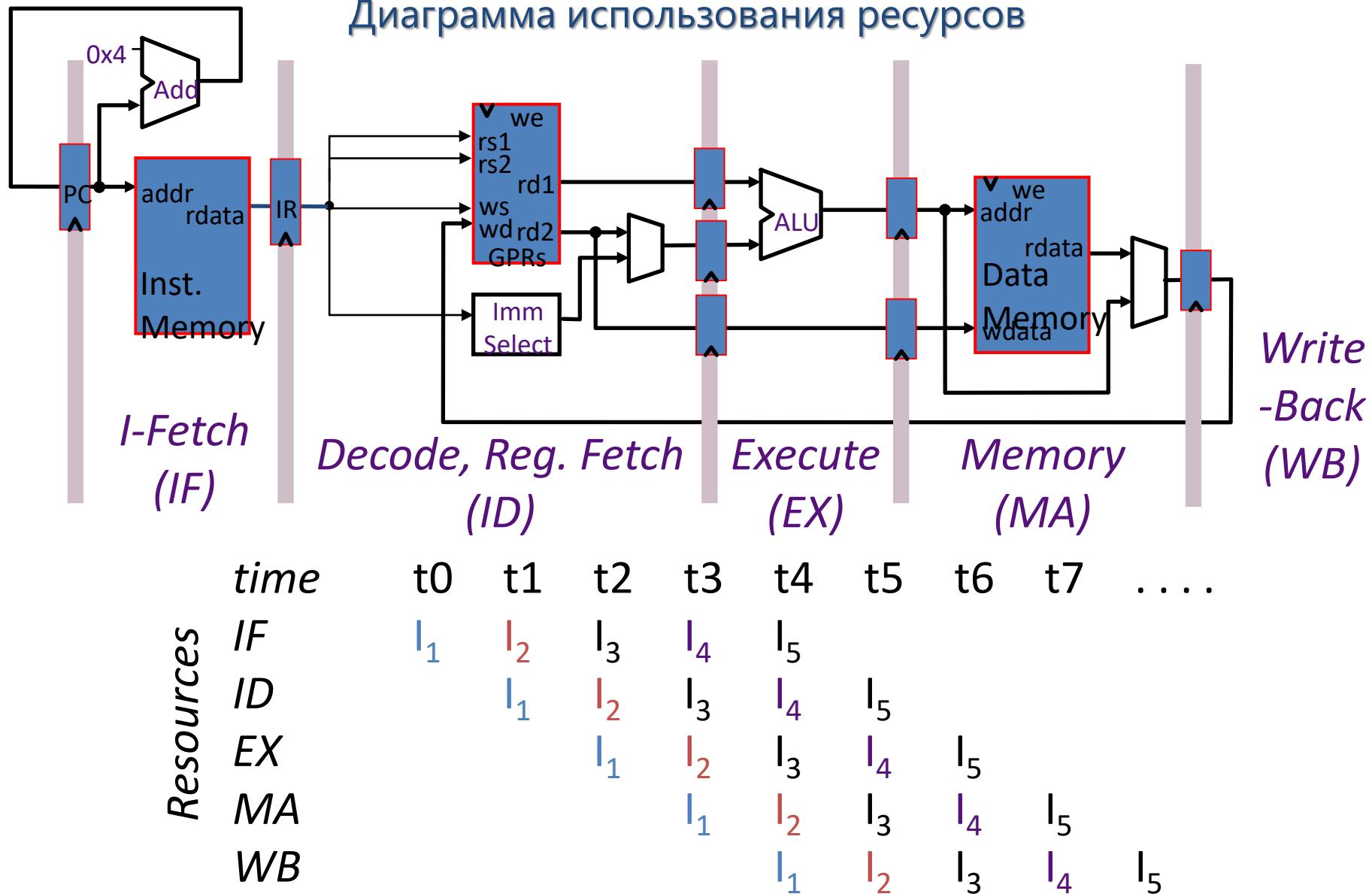
Пятистадийное конвейерное исполнение команд



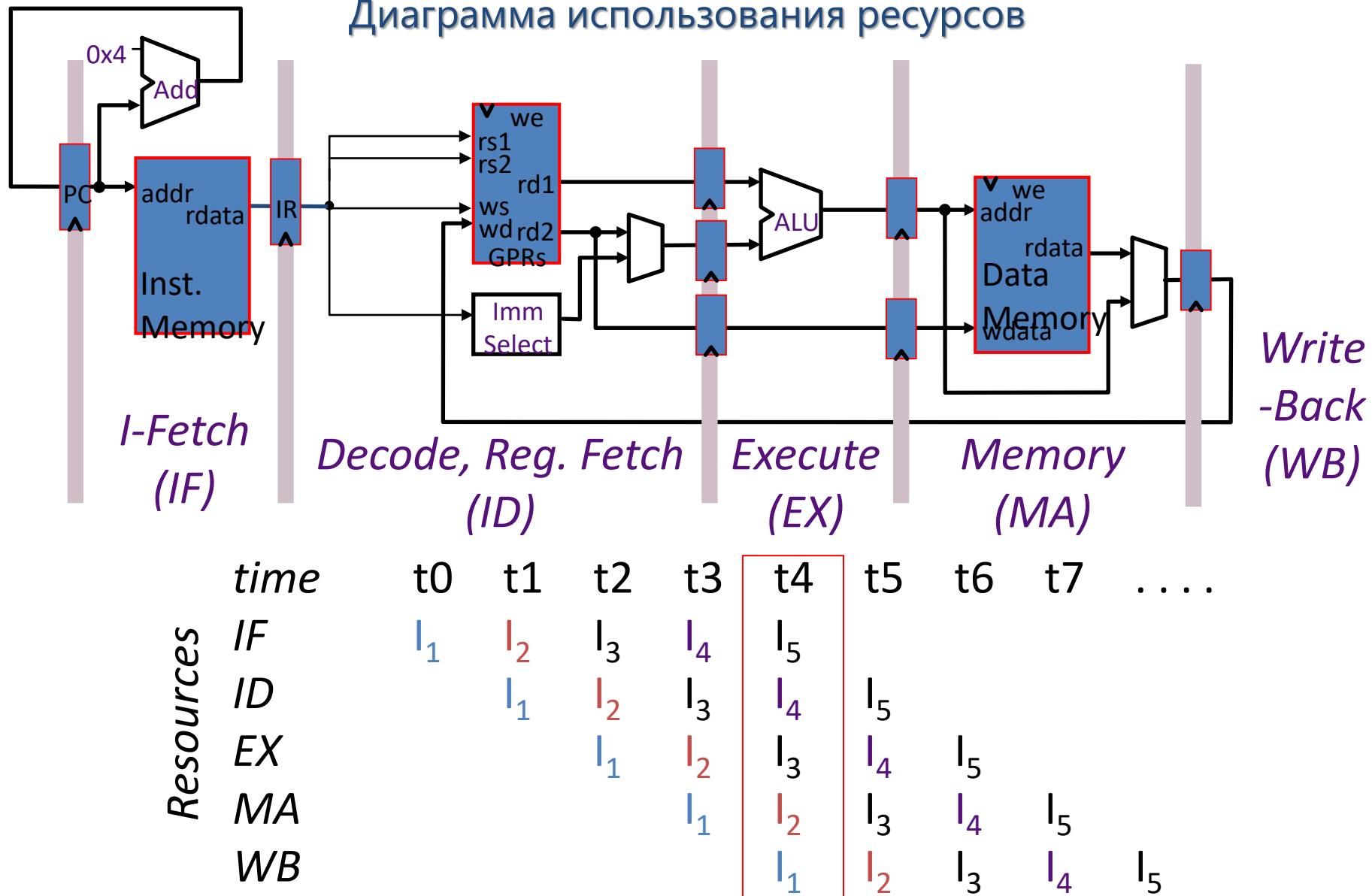
Пятистадийное конвейерное исполнение команд



Пятистадийное конвейерное исполнение команд

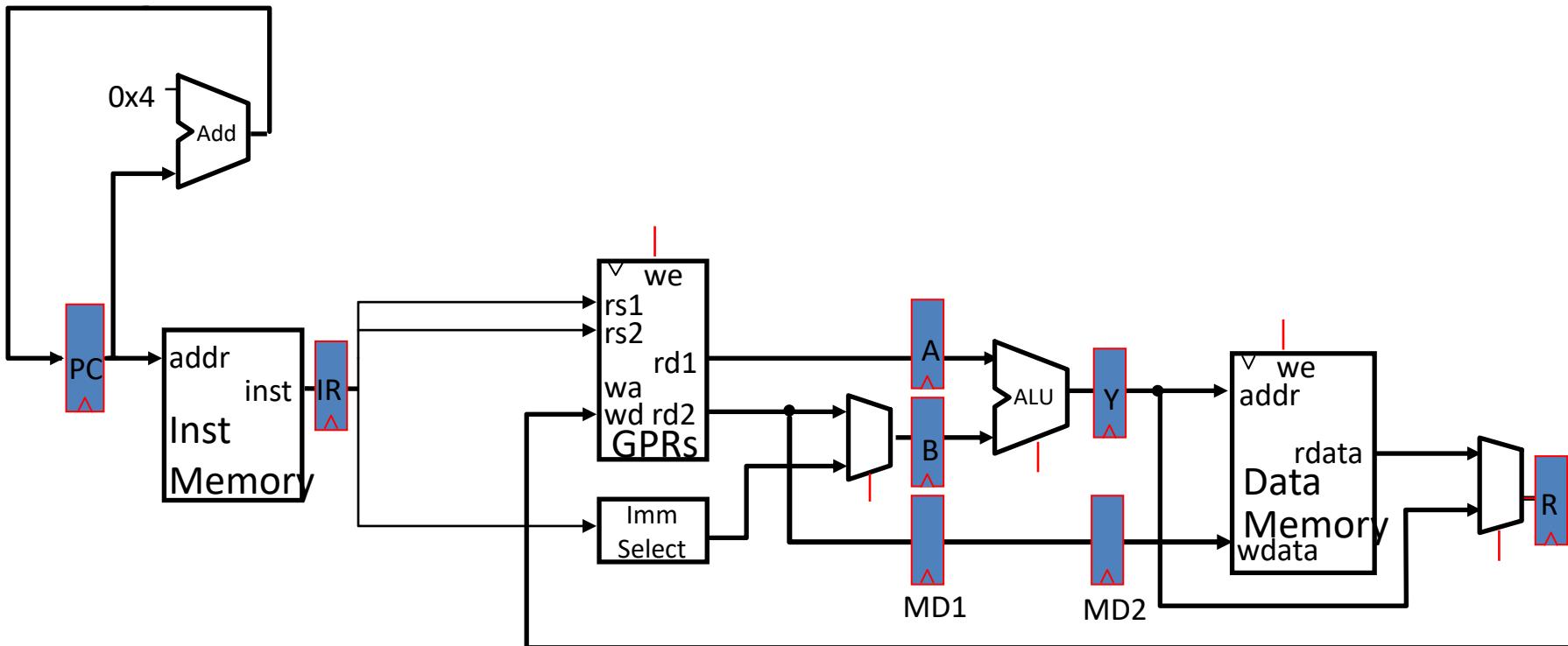


Пятистадийное конвейерное исполнение команд



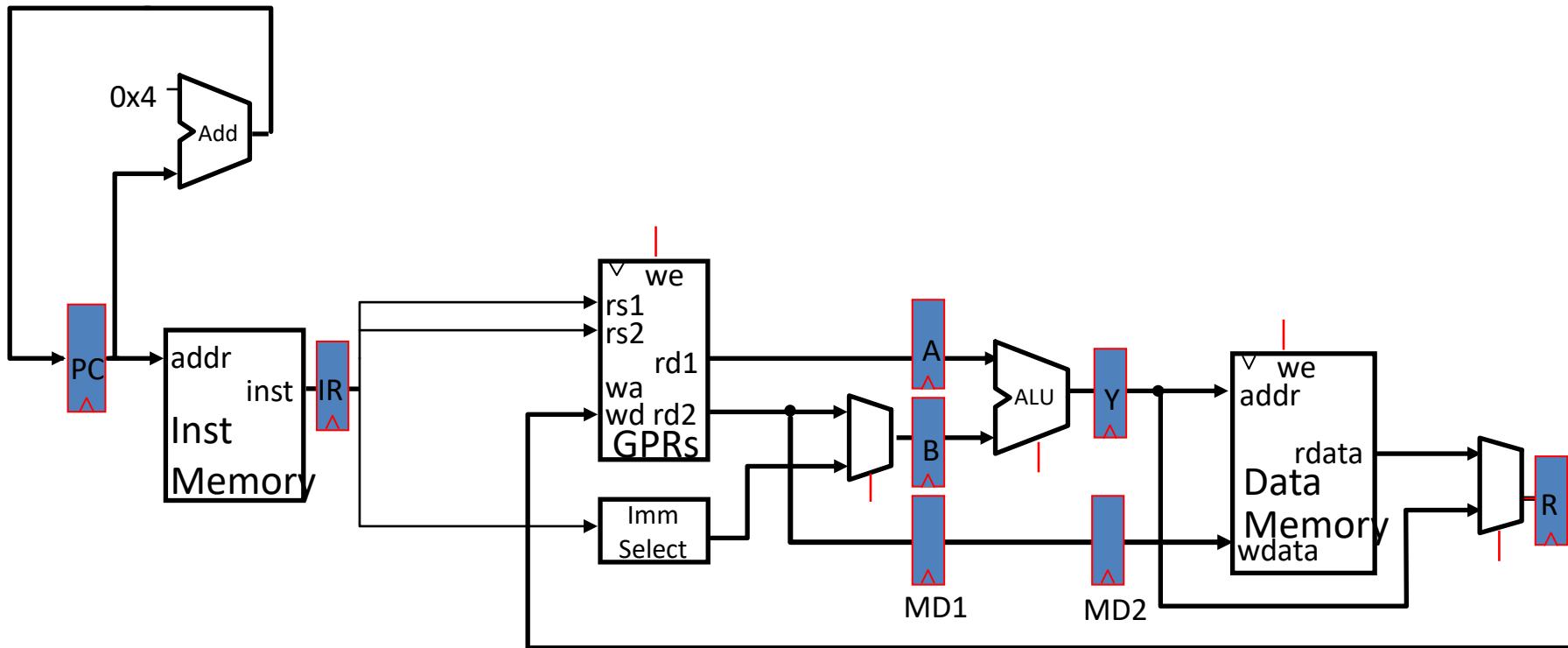
Конвейерное исполнение команд

ALU-инструкции



Конвейерное исполнение команд

ALU-инструкции

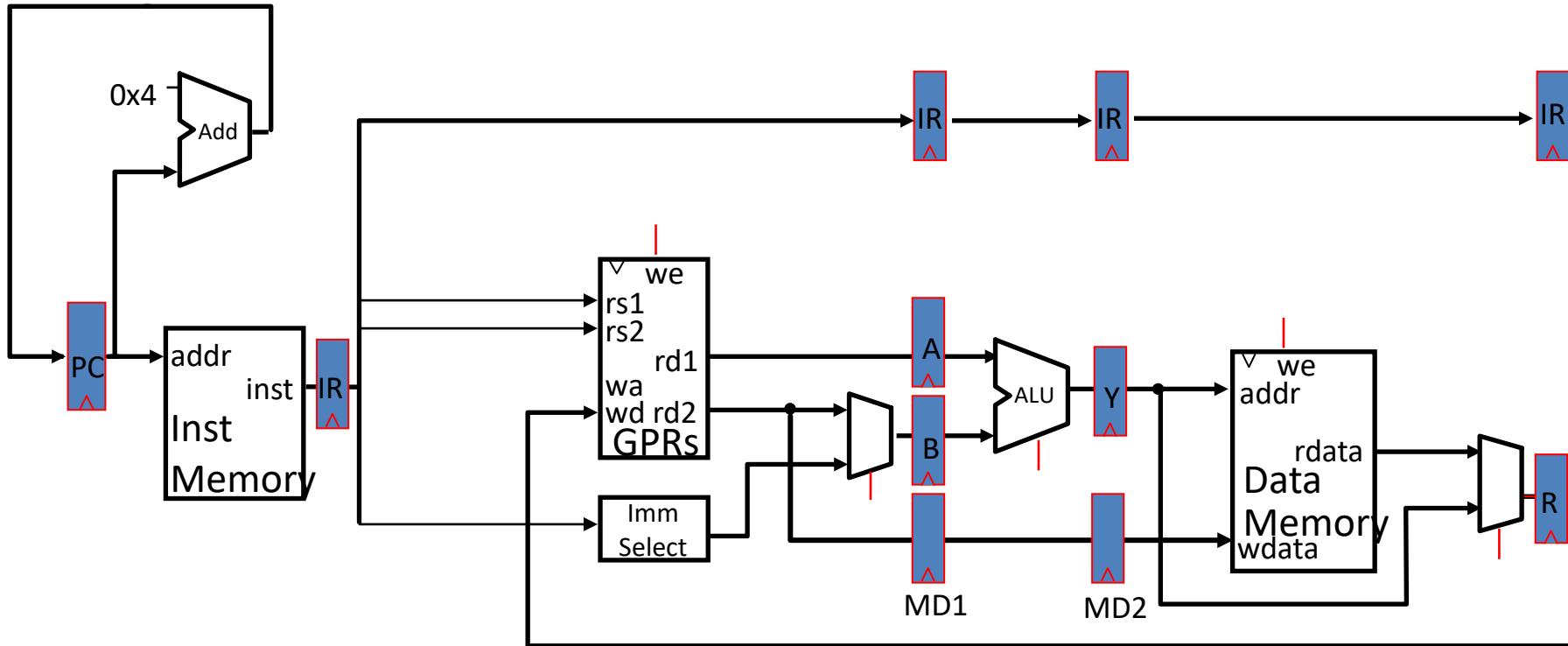


Не совсем верно!

Регистр инструкции (*Instruction Register, IR*) нужен для каждой стадии

Конвейерное исполнение команд

ALU-инструкции

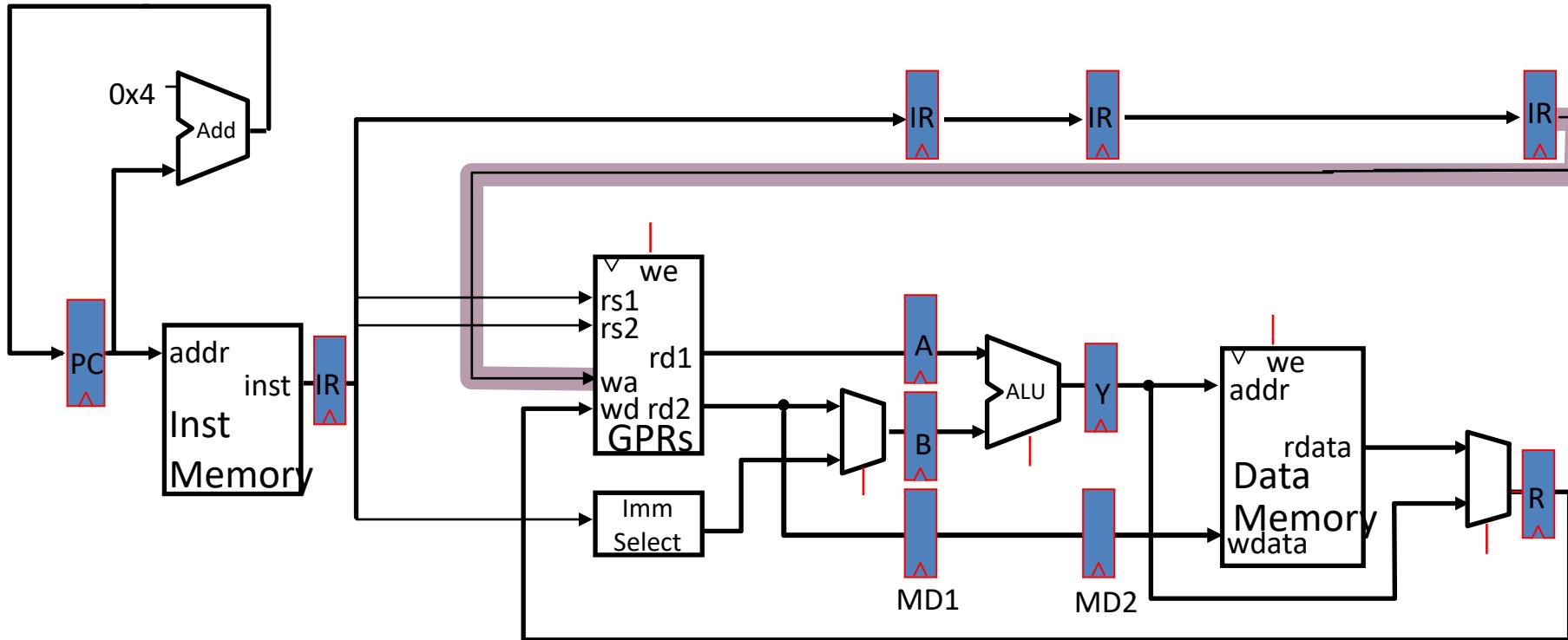


Не совсем верно!

Регистр инструкции (Instruction Register, IR) нужен для каждой стадии

Конвейерное исполнение команд

ALU-инструкции

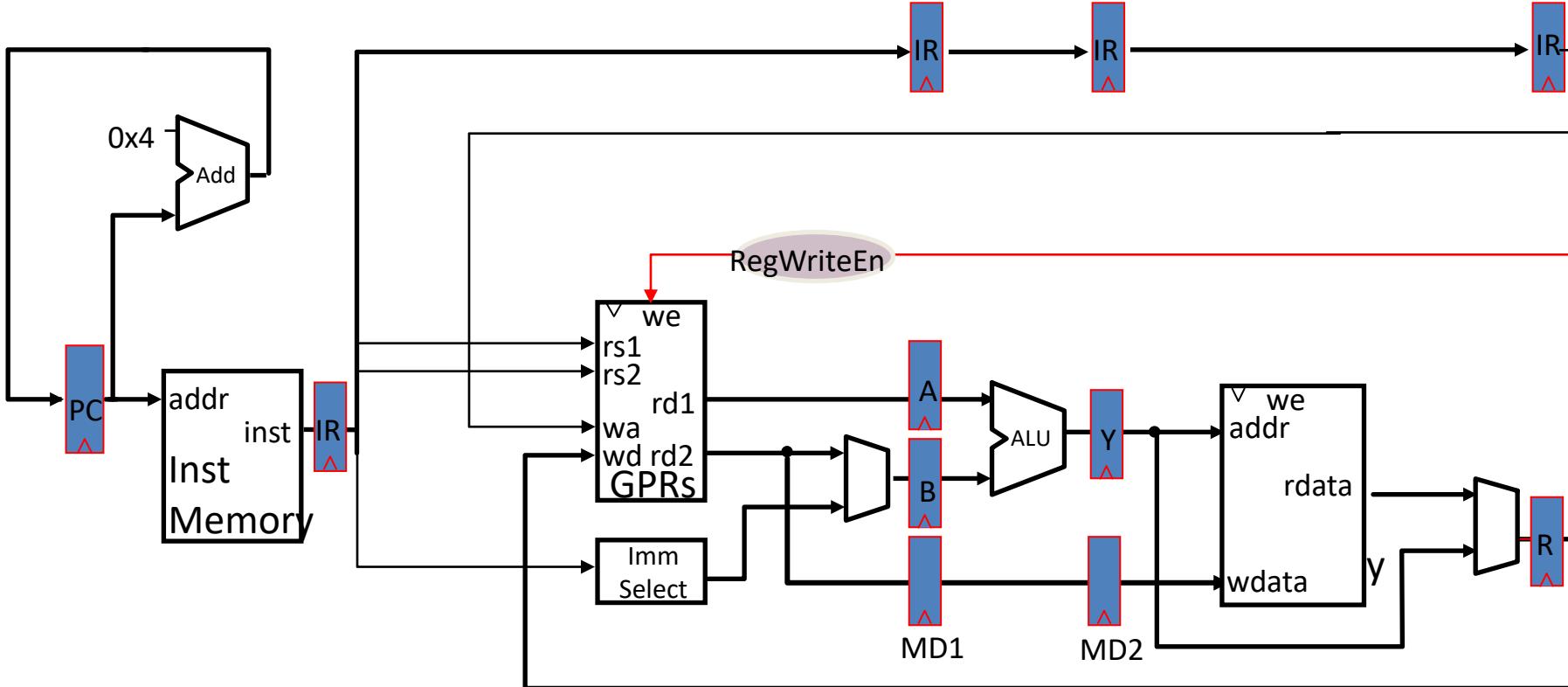


Не совсем верно!

Регистр инструкции (Instruction Register, IR) нужен для каждой стадии

Конвейеризованный тракт данных RISC-V

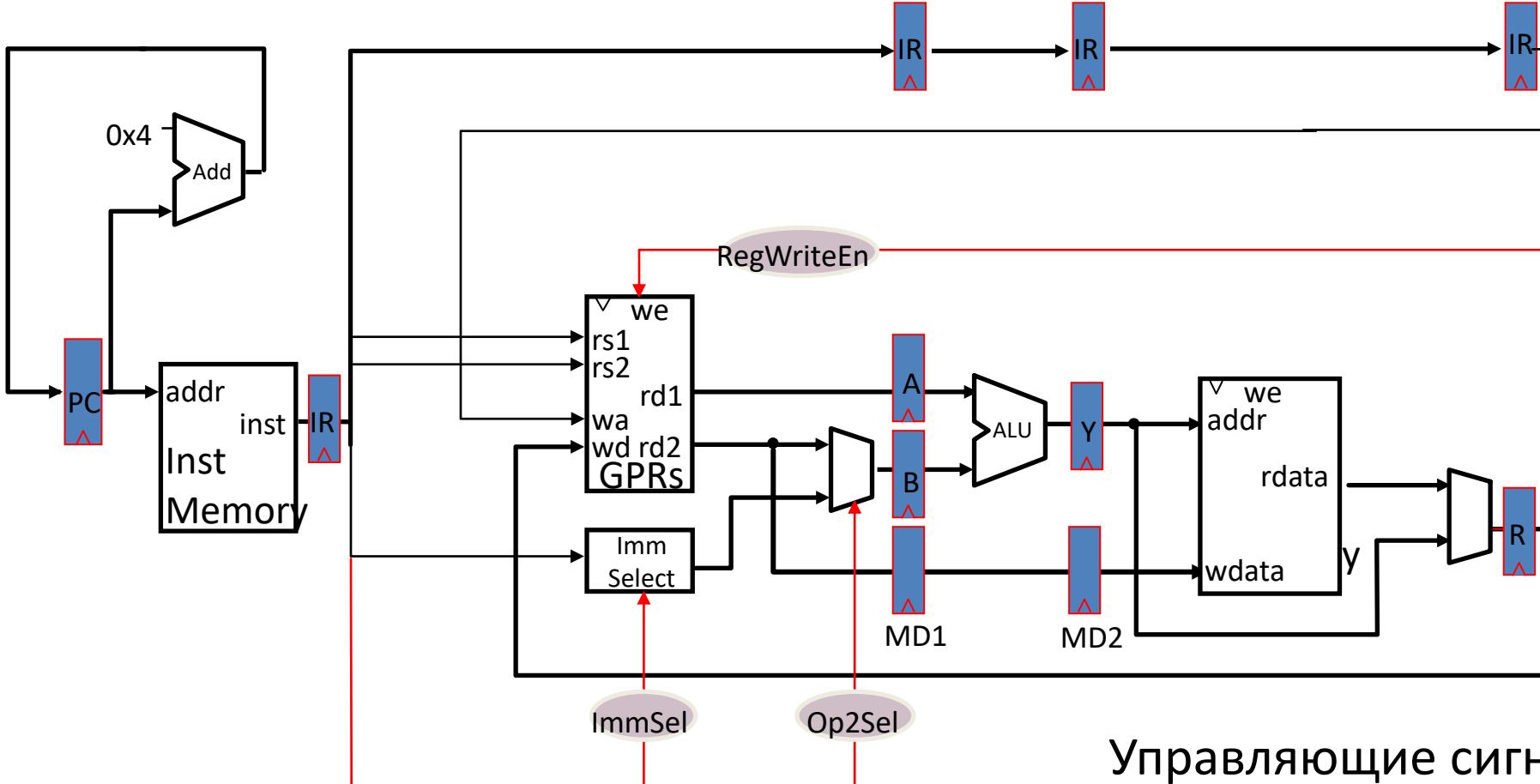
без инструкций безусловного перехода (跳跃)



Управляющие сигналы
должны быть объединены

Конвейеризованный тракт данных RISC-V

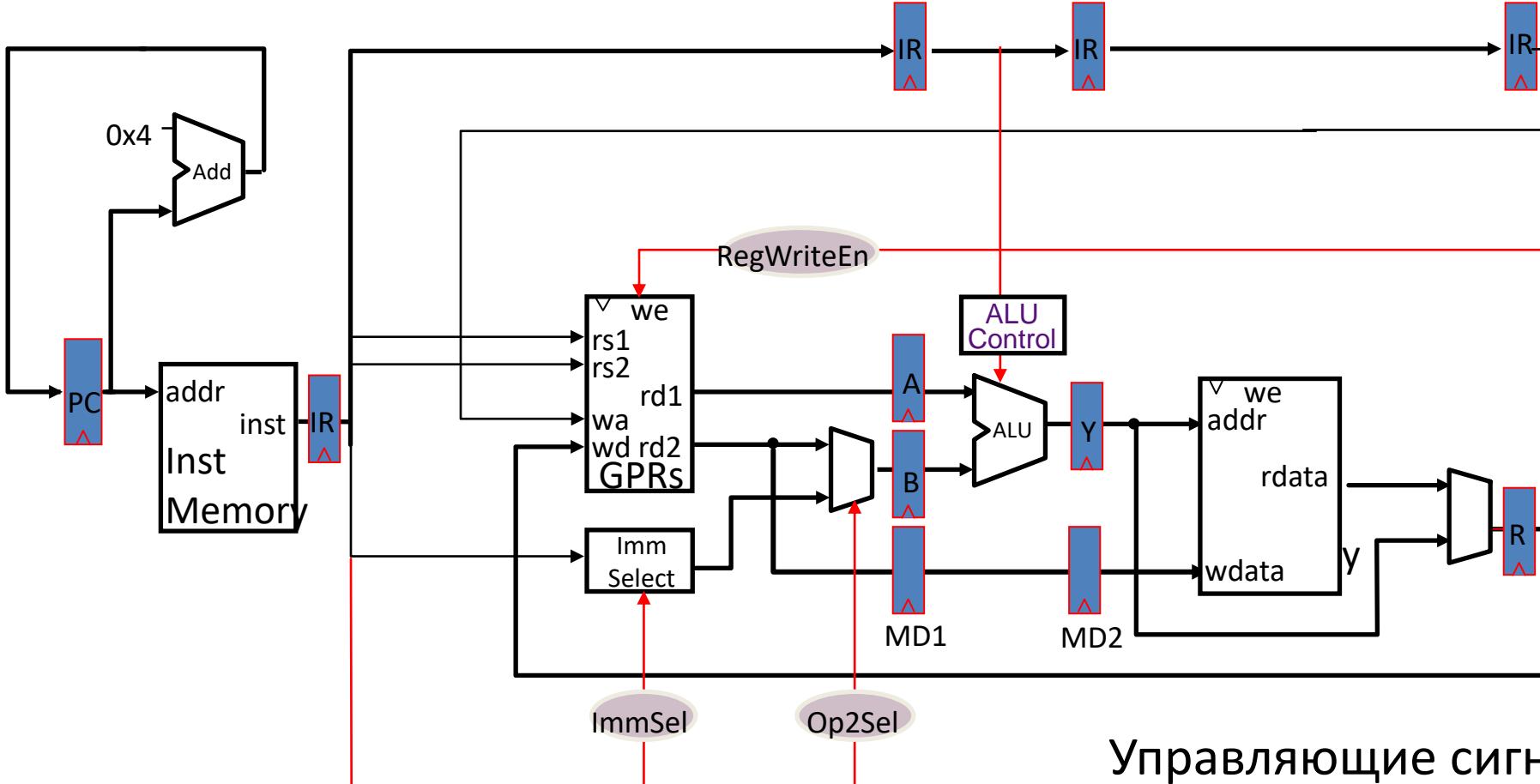
без инструкций безусловного перехода (jump)



Управляющие сигналы
должны быть объединены

Конвейеризованный тракт данных RISC-V

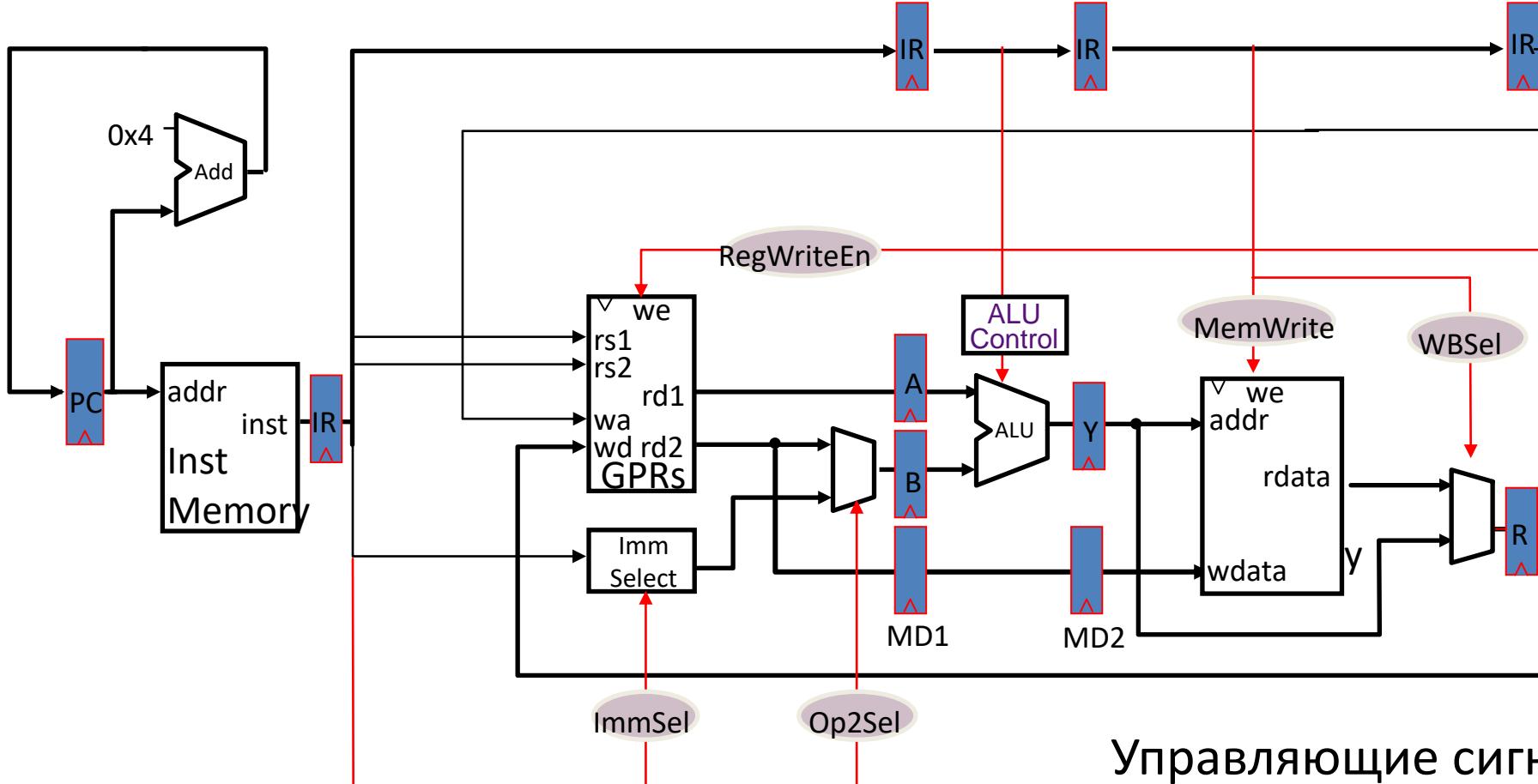
без инструкций безусловного перехода (跳跃)



Управляющие сигналы
должны быть объединены

Конвейеризованный тракт данных RISC-V

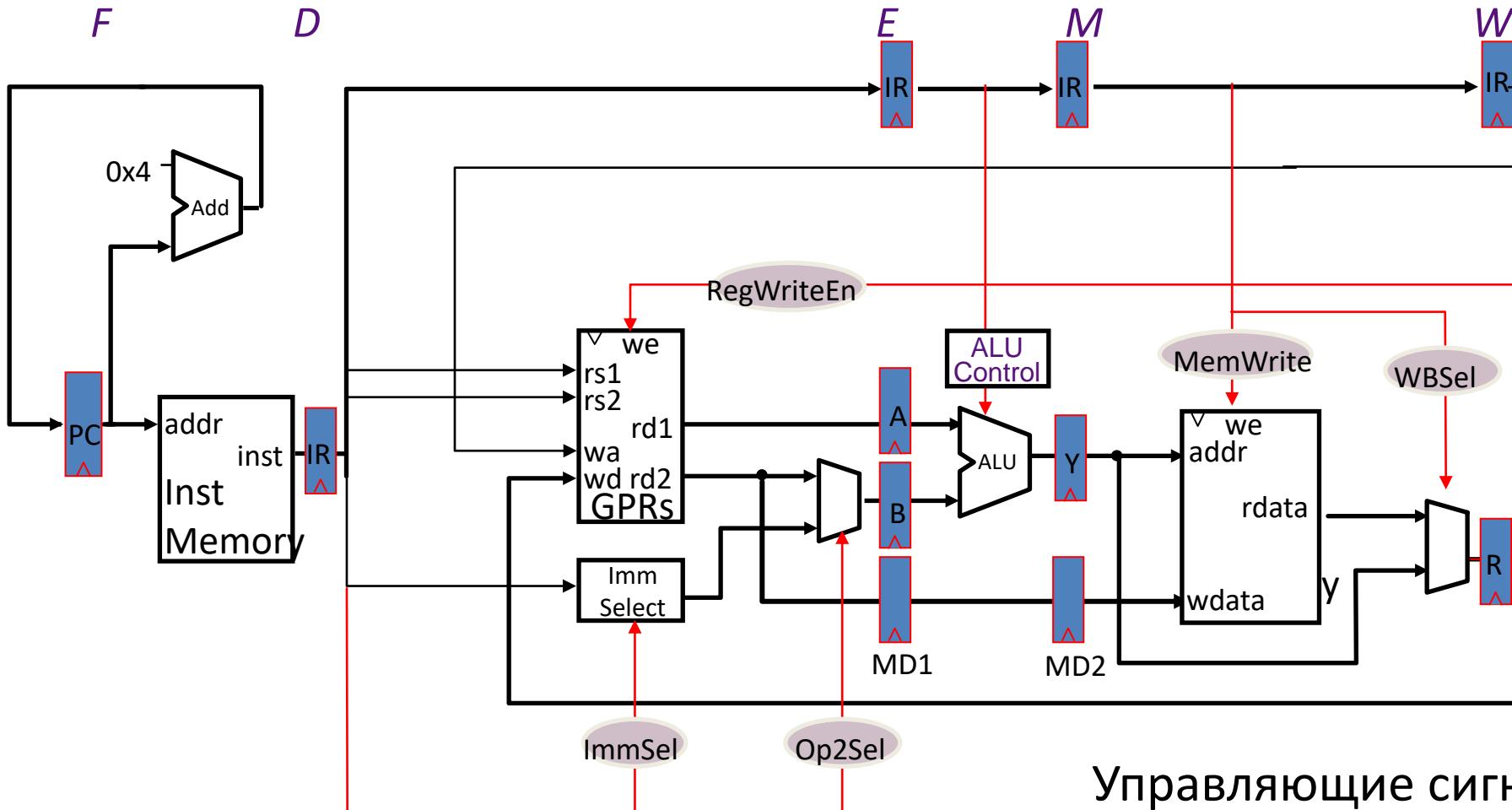
без инструкций безусловного перехода (跳跃)



Управляющие сигналы
должны быть объединены

Конвейеризованный тракт данных RISC-V

без инструкций безусловного перехода (jump)



Управляющие сигналы
должны быть объединены

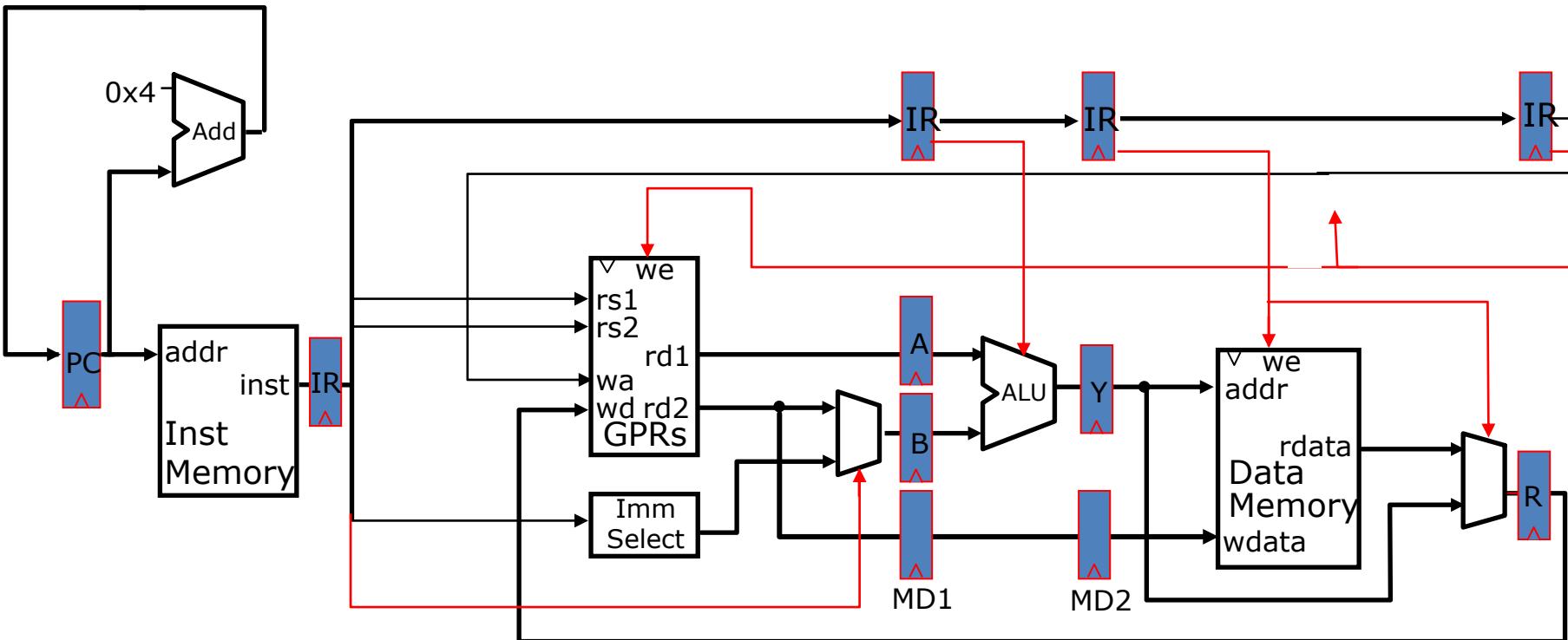
Коллизии на конвейере

- Структурная коллизия – командам необходимы некоторые ресурсы, занятые в данный момент другими командами – *structural hazard*
- Коллизии по данным – команда может вычислять данные или адрес, необходимые последующим инструкциям – *data hazard*
- Коллизии по управлению – команда может определять следующую исполняемую команду, в частности при переходе (branches, exceptions) – *control hazard*

Устранение структурных коллизий

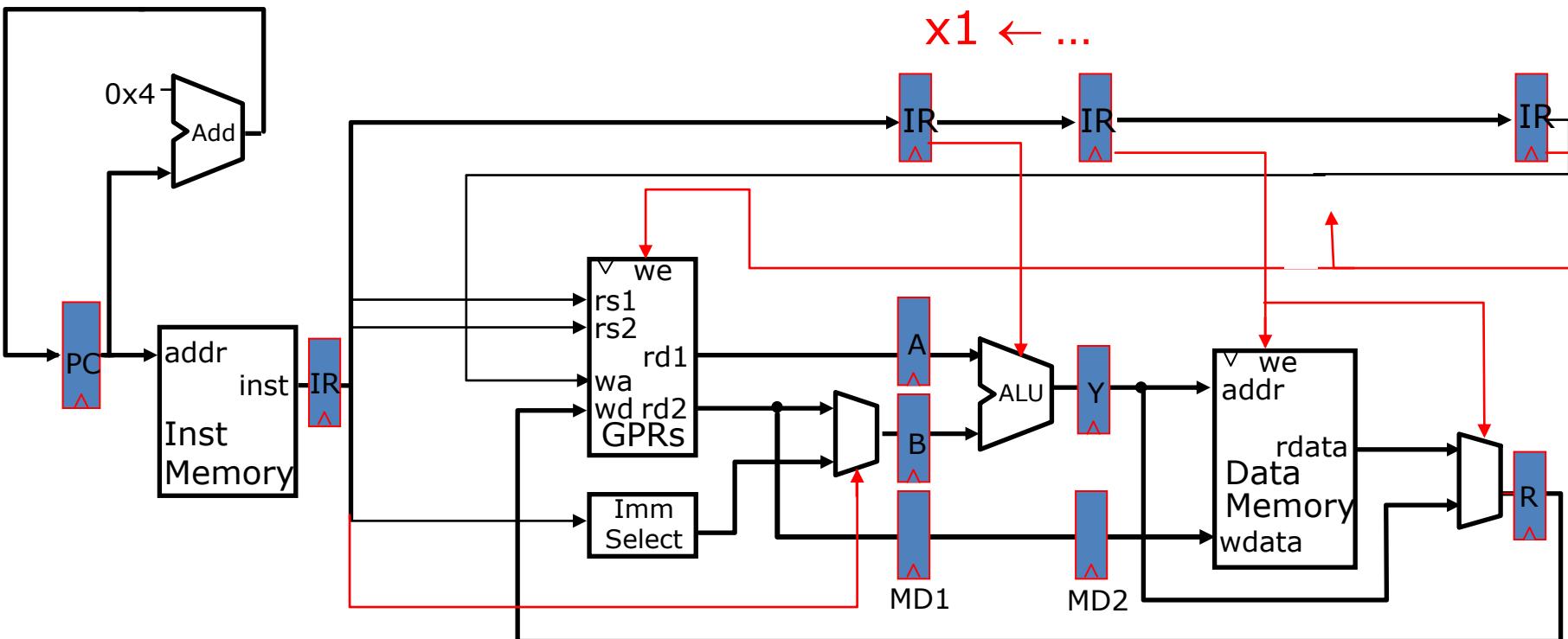
- Структурная коллизия происходит, когда двум инструкциям одновременно требуется один и тот же аппаратный ресурс
 - Можно устранить в аппаратуре путем блокировки (stall) новой инструкции вплоть до того, как старая инструкция закончит работу с ресурсом
- Структурную коллизию всегда можно устранить, добавляя больше аппаратуры
 - Например, если две инструкции одновременно запрашивают доступ в память, коллизия может быть устранена путем добавления второго порта по доступу к памяти
- В нашем пятистадийном конвейере по замыслу (by design) нет структурных коллизий

Коллизии по данным

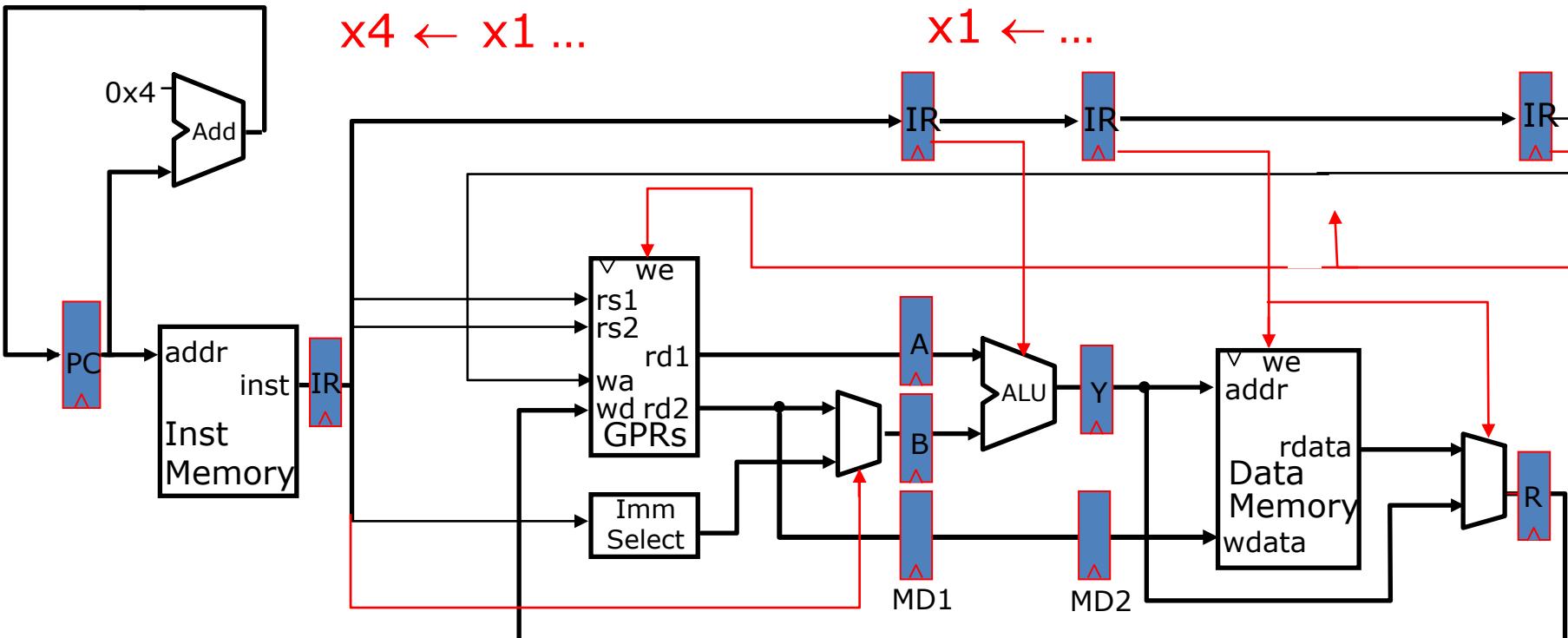


...
 $x_1 \leftarrow x_0 + 10$
 $x_4 \leftarrow x_1 + 17$
...

Коллизии по данным

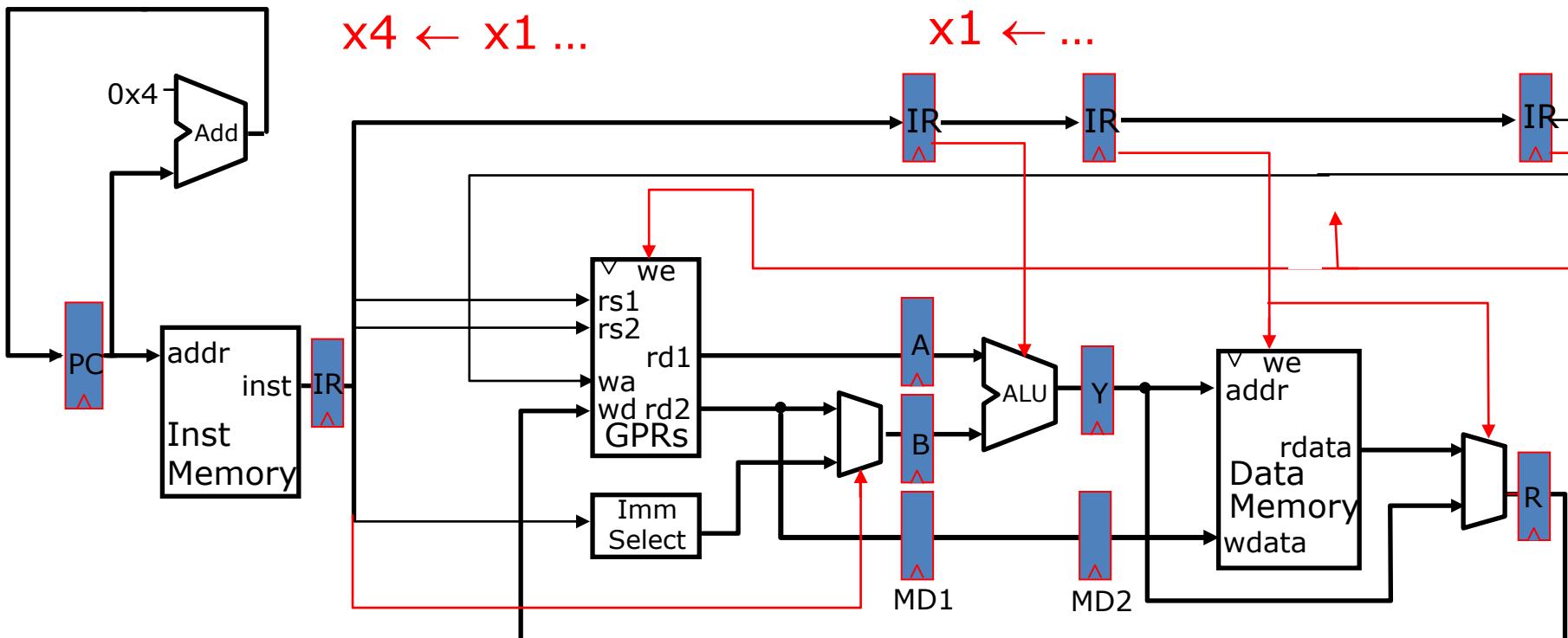


Коллизии по данным



...
 $x1 \leftarrow x0 + 10$
 $x4 \leftarrow x1 + 17$
...

Коллизии по данным



...
 $x1 \leftarrow x0 + 10$
 $x4 \leftarrow x1 + 17$
...

Значение $x1$ устаревшее

Как можно устранить такую коллизию?

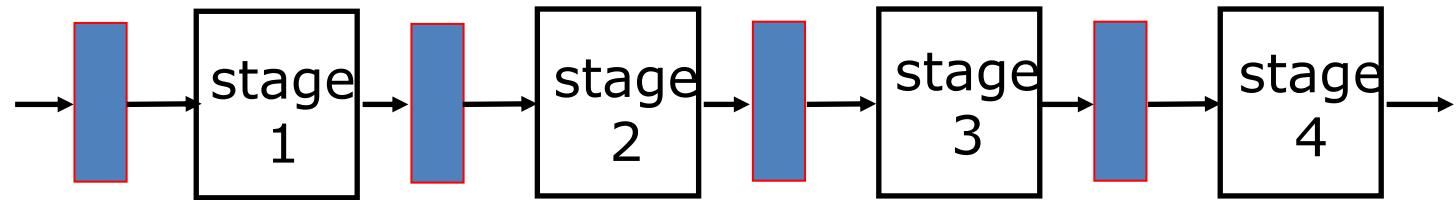
- Три опции:
 - Ожидание (блокировка, stall)
 - Предсказание значения
 - Байпас: запросить значение до того, как оно будет записано в память

Устранение коллизий по данным (1)

Стратегия 1:

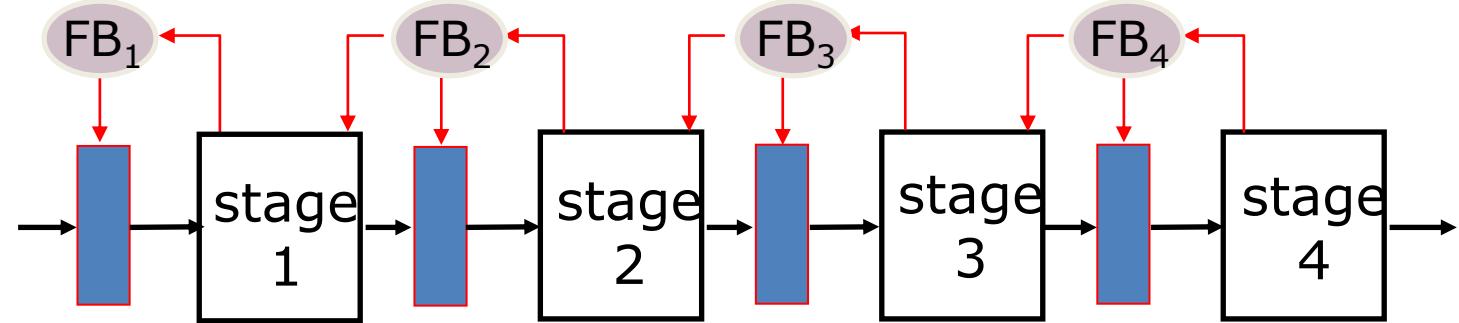
*Ожидание, когда результат будет получен, путем блокировки более ранних стадий конвейера → *interlocks**

Фидбэк для устранения коллизий



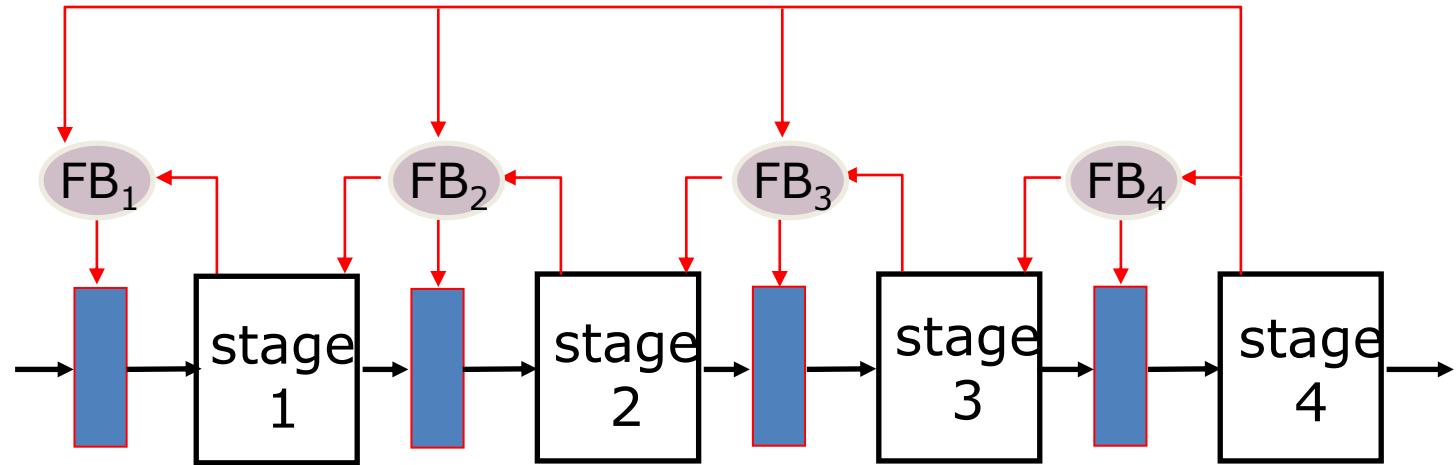
- Следующие стадии могут передавать информацию о зависимостях на предыдущие стадии, которая позволяет *блокировать* или «убить» инструкцию

Фидбэк для устранения коллизий



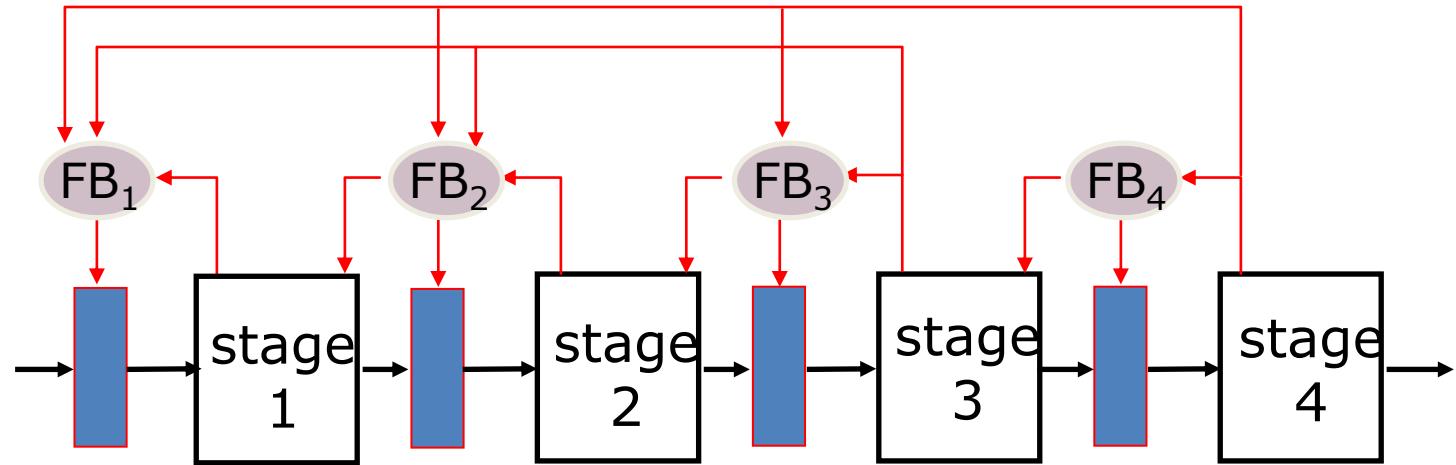
- Следующие стадии могут передавать информацию о зависимостях на предыдущие стадии, которая позволяет *блокировать* или «убить» инструкцию

Фидбэк для устранения коллизий



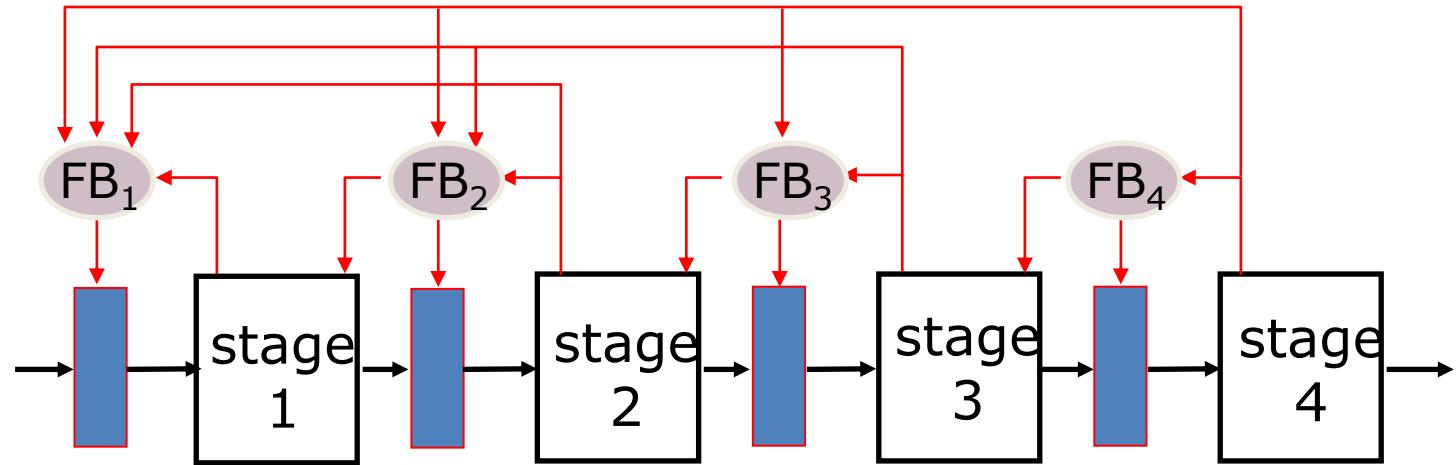
- Следующие стадии могут передавать информацию о зависимостях на предыдущие стадии, которая позволяет *блокировать* или «убить» инструкцию

Фидбэк для устранения коллизий



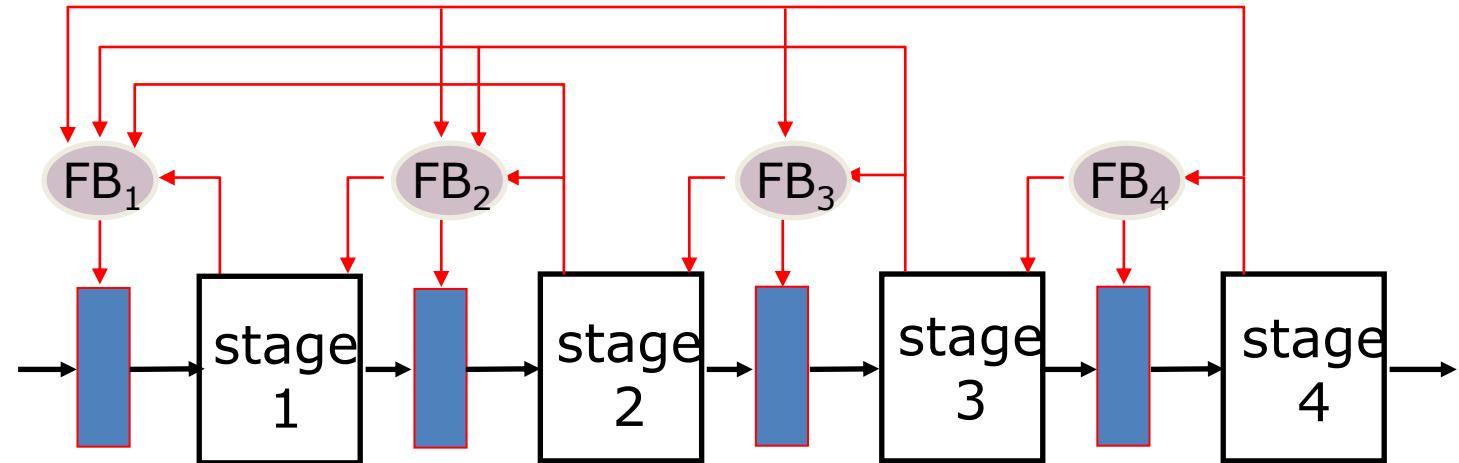
- Следующие стадии могут передавать информацию о зависимостях на предыдущие стадии, которая позволяет *блокировать* или «убить» инструкцию

Фидбэк для устранения коллизий



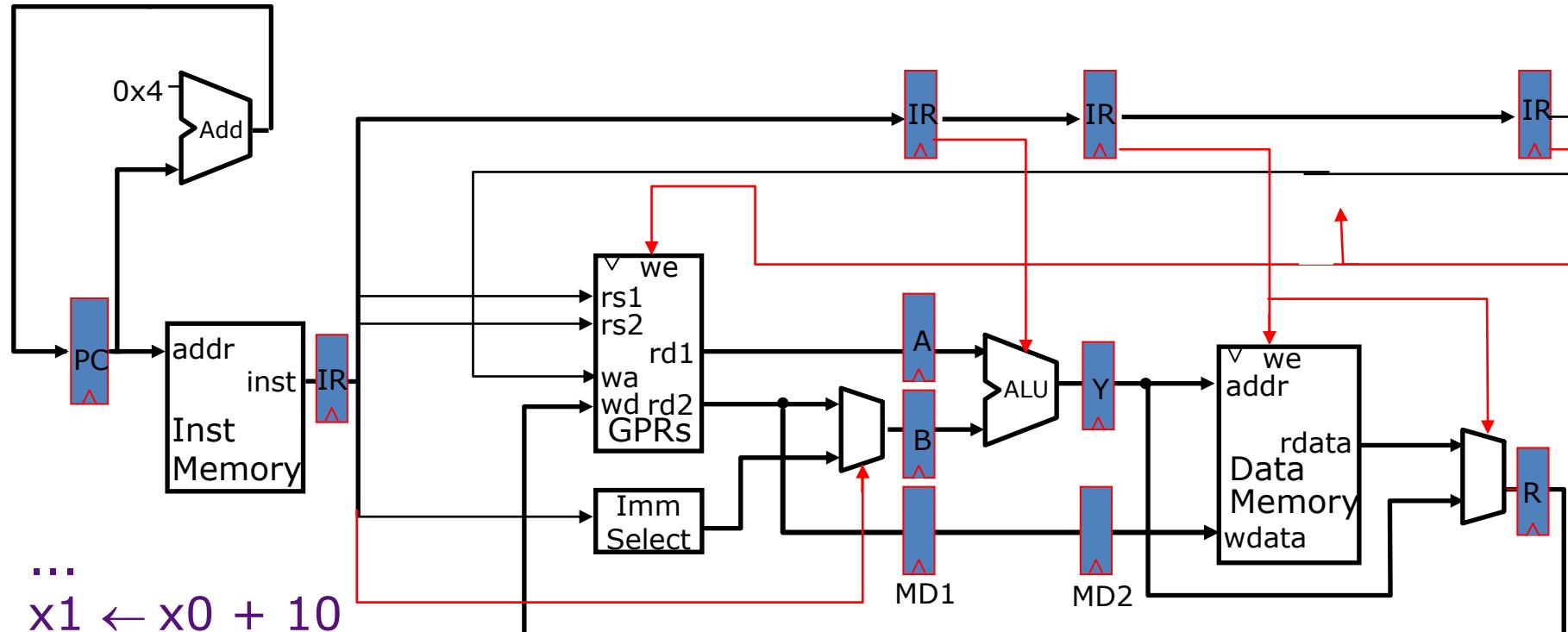
- Следующие стадии могут передавать информацию о зависимостях на предыдущие стадии, которая позволяет *блокировать* или «убить» инструкцию

Фидбэк для устранения коллизий



- Следующие стадии могут передавать информацию о зависимостях на предыдущие стадии, которая позволяет блокировать или «убить» инструкцию
- Управление конвейером таким способом работает только в том случае, если инструкция на стадии $i+1$ может завершиться без какого-либо вмешательства со стороны инструкций на стадиях от 1 до i (иначе могут случиться deadlock'и)

Interlock'и для устранения коллизий по данным

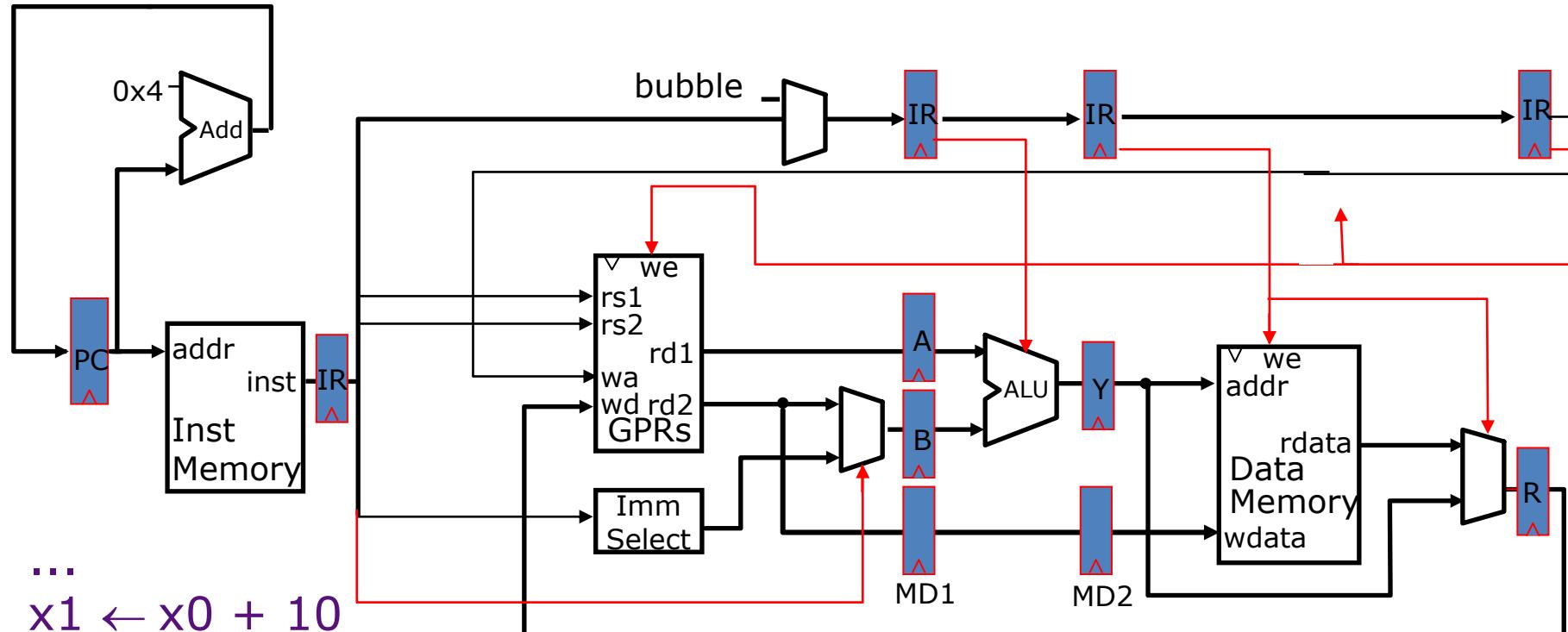


$$x_1 \leftarrow x_0 + 10$$

$$x_4 \leftarrow x_1 + 17$$

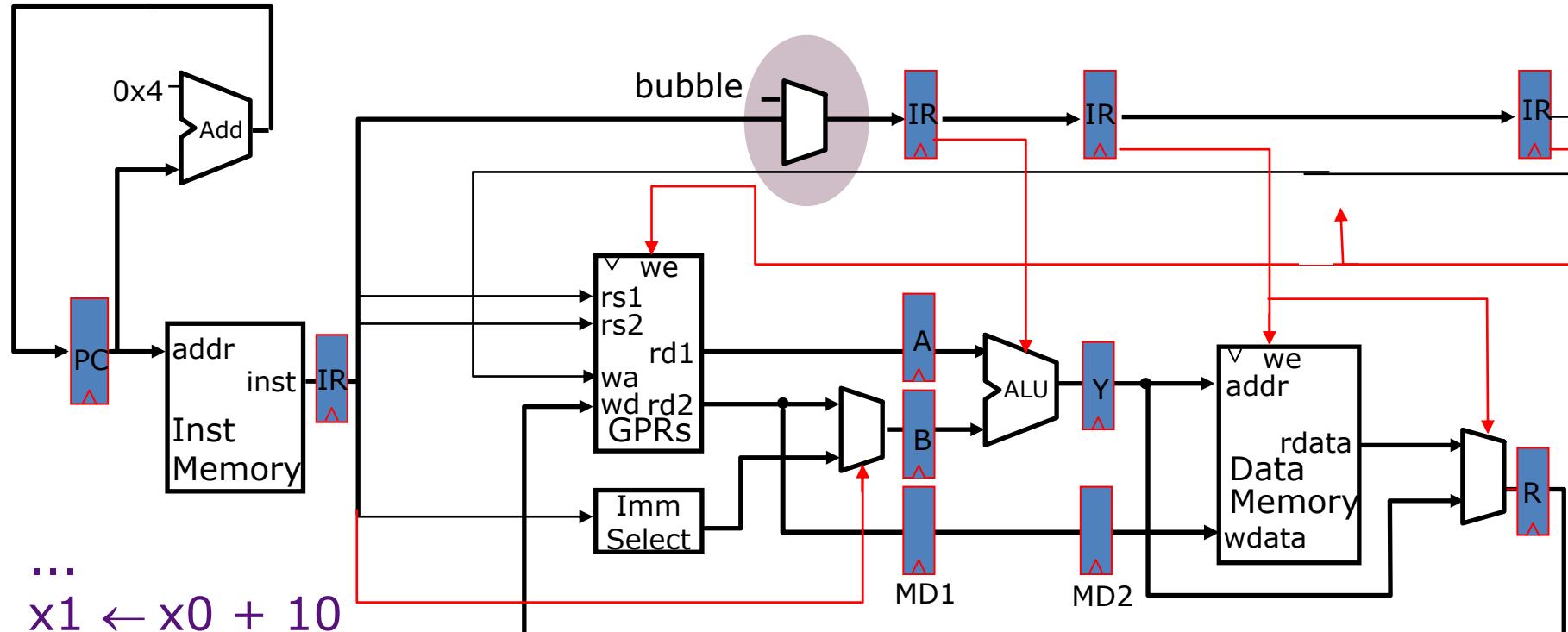
...

Interlock'и для устранения коллизий по данным



$x_1 \leftarrow x_0 + 10$
 $x_4 \leftarrow x_1 + 17$
...

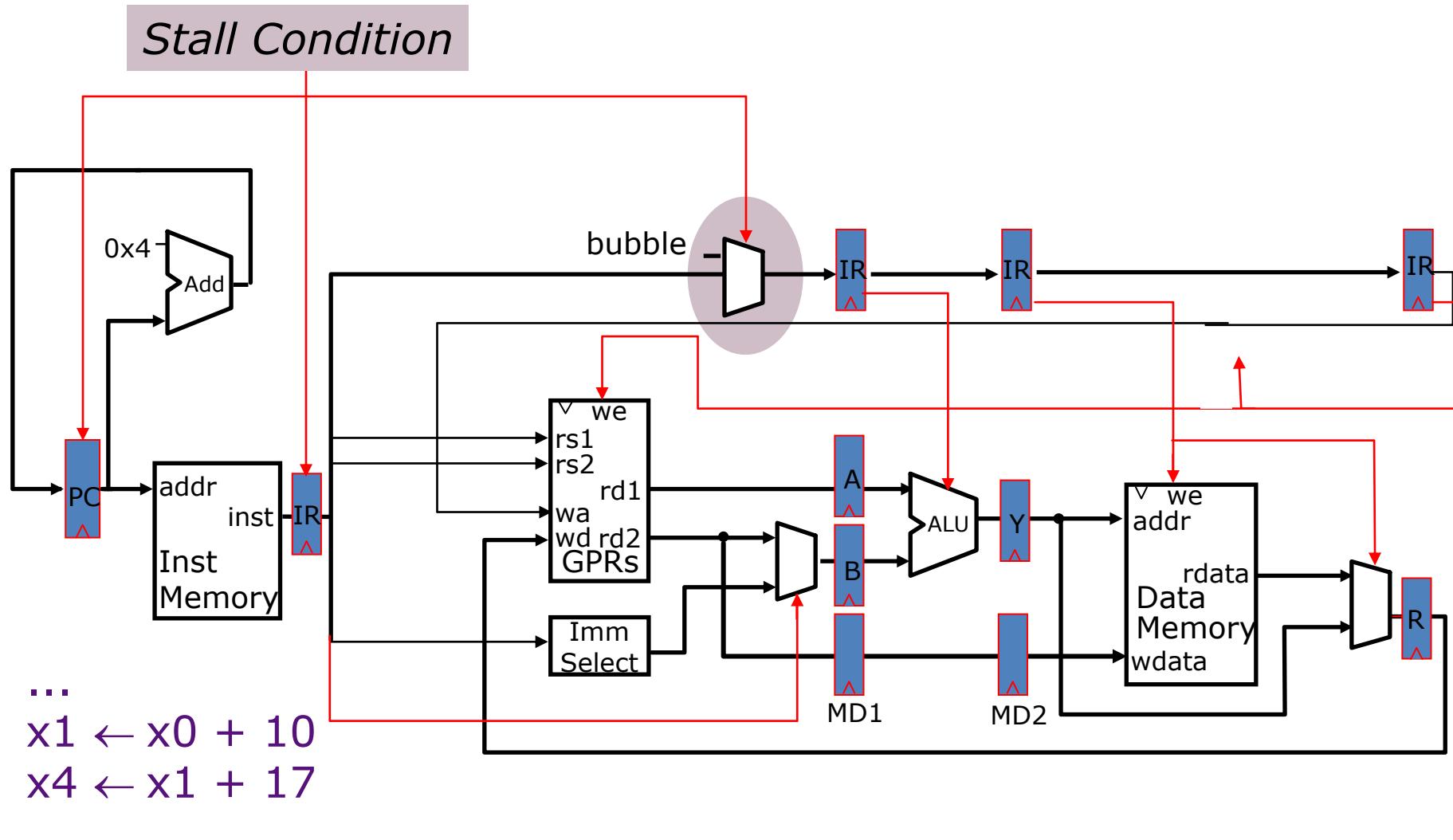
Interlock'и для устранения коллизий по данным



$x_1 \leftarrow x_0 + 10$
 $x_4 \leftarrow x_1 + 17$

...

Interlock'и для устранения коллизий по данным



Блокированные стадии и конвейерные «пузырьки»

time
t0 t1 t2 t3 t4 t5 t6 t7 . . .

Блокированные стадии и конвейерные «пузырьки»

	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) x1 ← (x0)+10	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				

Блокированные стадии и конвейерные «пузырьки»

	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) x1 ← (x0)+10	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) x4 ← (x1)+17		IF ₂	ID ₂	ID ₂	ID ₂	EX ₂	MA ₂	WB ₂	

Блокированные стадии и конвейерные «пузырьки»

	<i>time</i>									
	t0	t1	t2	t3	t4	t5	t6	t7	...	
(I ₁) x1 ← (x0)+10	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁					
(I ₂) x4 ← (x1)+17		IF ₂	ID ₂	ID ₂	ID ₂	EX ₂	MA ₂	WB ₂		
(I ₃)			IF ₃	IF ₃	IF ₃	ID ₃	EX ₃	MA ₃	WB ₃	

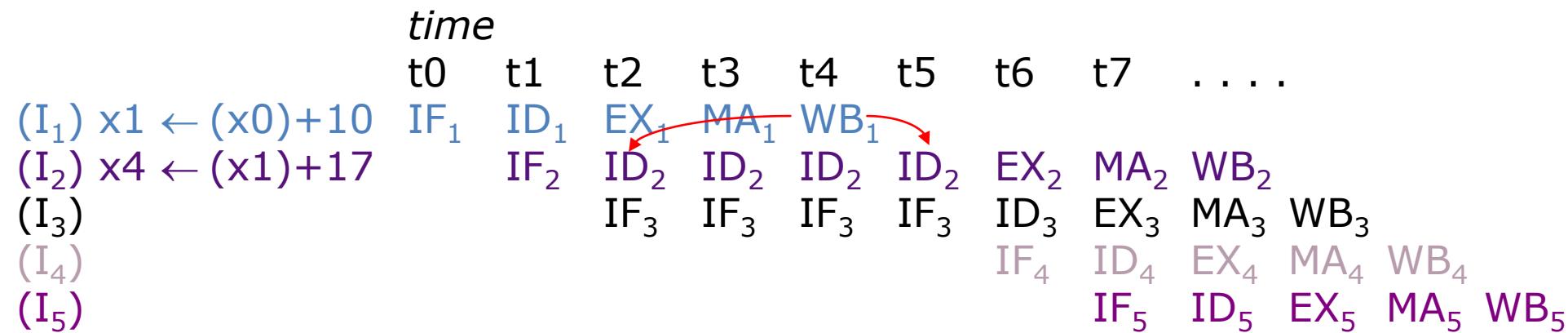
Блокированные стадии и конвейерные «пузырьки»

	time									
	t0	t1	t2	t3	t4	t5	t6	t7	...	
(I ₁) $x1 \leftarrow (x0) + 10$	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁					
(I ₂) $x4 \leftarrow (x1) + 17$		IF ₂	ID ₂	ID ₂	ID ₂	ID ₂	EX ₂	MA ₂	WB ₂	
(I ₃)			IF ₃	IF ₃	IF ₃	IF ₃	ID ₃	EX ₃	MA ₃	WB ₃
(I ₄)							IF ₄	ID ₄	EX ₄	MA ₄
										WB ₄

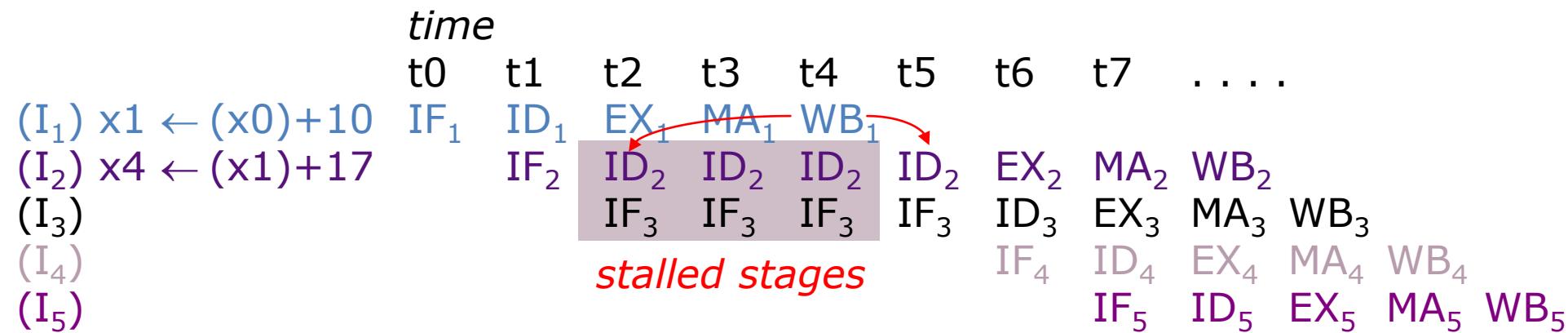
БЛОКИРОВАННЫЕ СТАДИИ И КОНВЕЙЕРНЫЕ «ПУЗЫРЬКИ»

	time										
	t0	t1	t2	t3	t4	t5	t6	t7	...		
(I ₁) $x_1 \leftarrow (x_0) + 10$	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁						
(I ₂) $x_4 \leftarrow (x_1) + 17$		IF ₂	ID ₂	ID ₂	ID ₂	ID ₂	EX ₂	MA ₂	WB ₂		
(I ₃)			IF ₃	IF ₃	IF ₃	IF ₃	ID ₃	EX ₃	MA ₃	WB ₃	
(I ₄)							IF ₄	ID ₄	EX ₄	MA ₄	WB ₄
(I ₅)							IF ₅	ID ₅	EX ₅	MA ₅	WB ₅

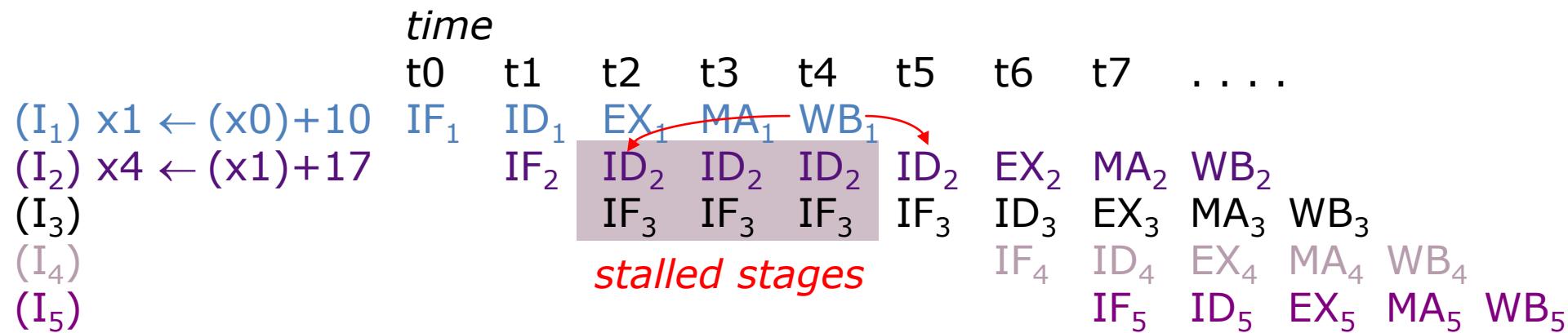
Блокированные стадии и конвейерные «пузырьки»



БЛОКИРОВАННЫЕ СТАДИИ И КОНВЕЙЕРНЫЕ «ПУЗЫРЬКИ»

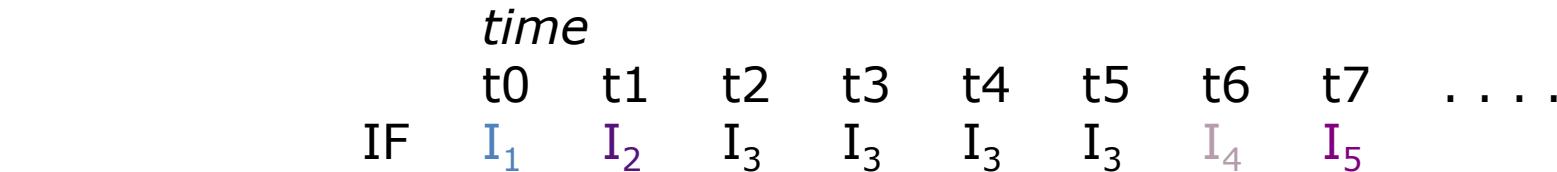
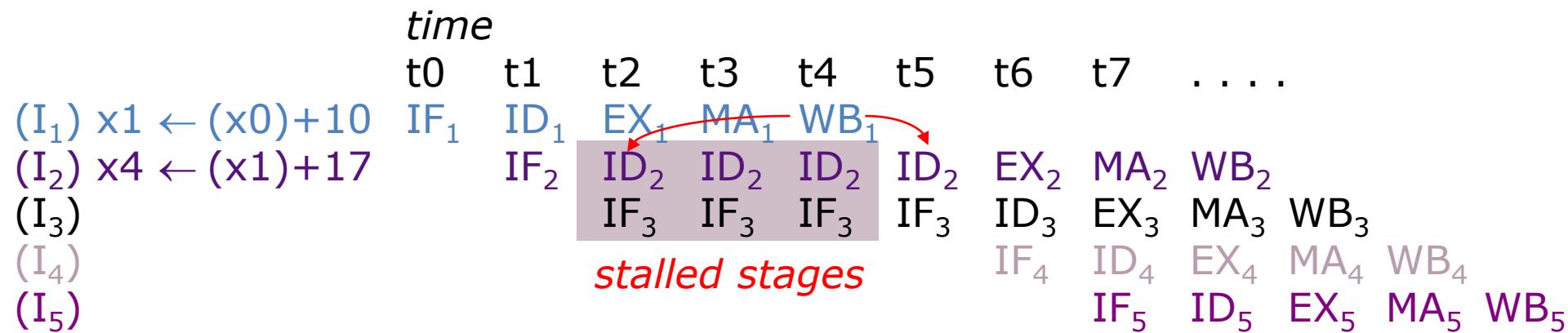


БЛОКИРОВАННЫЕ СТАДИИ И КОНВЕЙЕРНЫЕ «ПУЗЫРЬКИ»



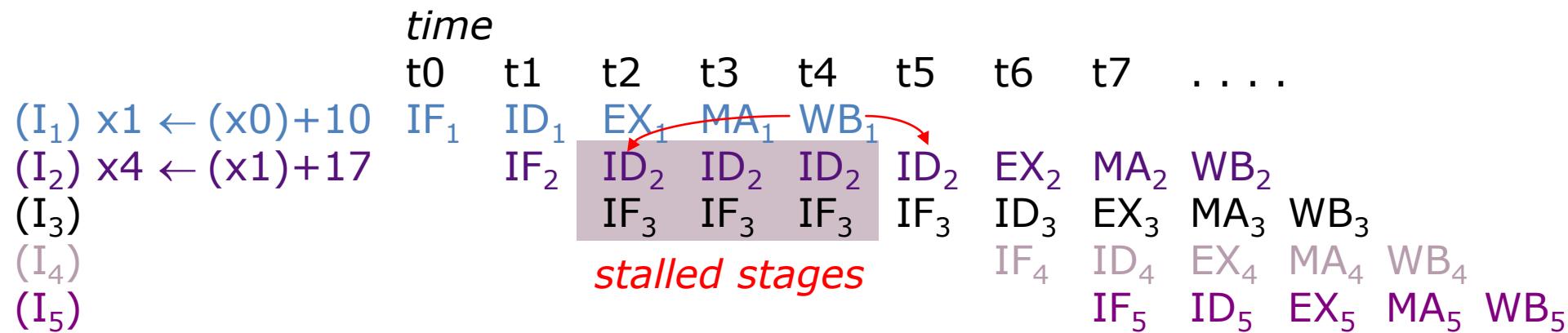
*Resource
Usage*

БЛОКИРОВАННЫЕ СТАДИИ И КОНВЕЙЕРНЫЕ «ПУЗЫРЬКИ»



*Resource
Usage*

БЛОКИРОВАННЫЕ СТАДИИ И КОНВЕЙЕРНЫЕ «ПУЗЫРЬКИ»

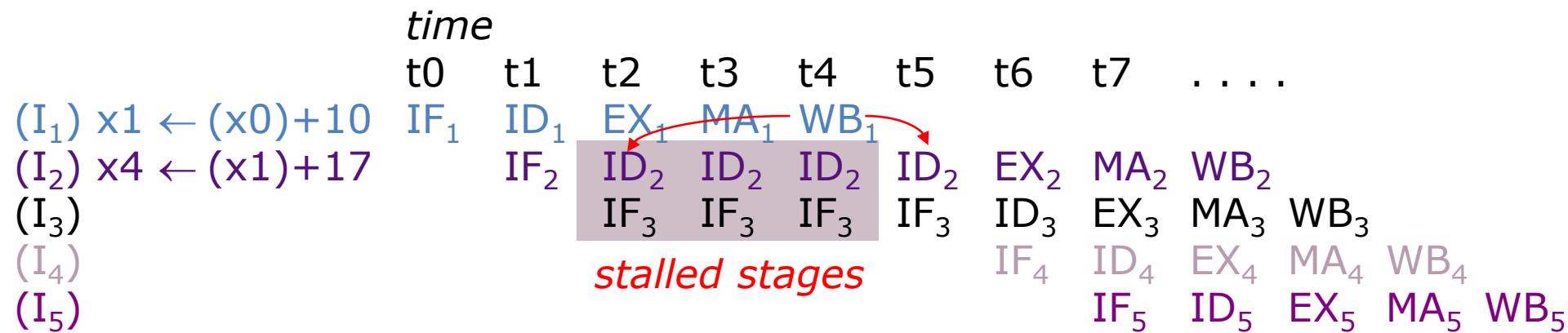


time

	t0	t1	t2	t3	t4	t5	t6	t7	...
IF	I_1	I_2	I_3	I_3	I_3	I_3	I_4	I_5	
ID		I_1	I_2	I_2	I_2	I_2	I_3	I_4	I_5

Resource Usage

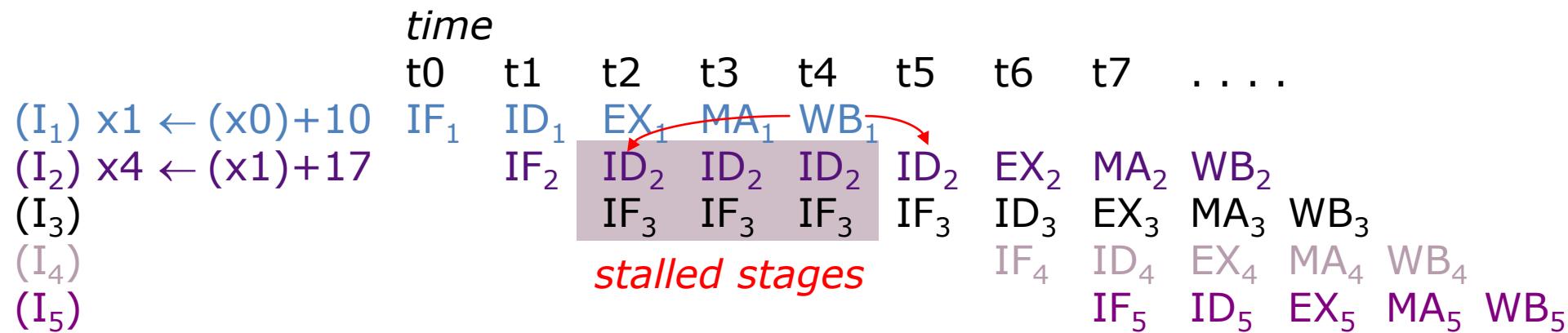
БЛОКИРОВАННЫЕ СТАДИИ И КОНВЕЙЕРНЫЕ «ПУЗЫРЬКИ»



time

	t0	t1	t2	t3	t4	t5	t6	t7	...
Resource Usage	IF	I ₁	I ₂	I ₃	I ₃	I ₃	I ₄	I ₅	
	ID		I ₁	I ₂	I ₂	I ₂	I ₃	I ₄	I ₅
	EX			I ₁	-	-	I ₂	I ₃	I ₄

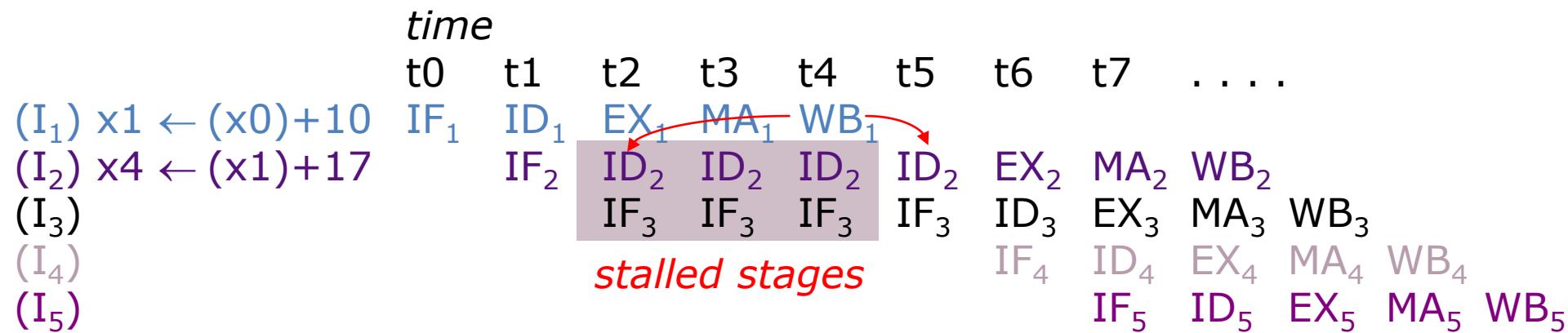
БЛОКИРОВАННЫЕ СТАДИИ И КОНВЕЙЕРНЫЕ «ПУЗЫРЬКИ»



time

	t0	t1	t2	t3	t4	t5	t6	t7	...
Resource Usage	IF	I_1	I_2	I_3	I_3	I_3	I_4	I_5	
	ID		I_1	I_2	I_2	I_2	I_3	I_4	I_5
	EX			I_1	-	-	I_2	I_3	I_4
	MA				I_1	-	-	I_2	I_3

БЛОКИРОВАННЫЕ СТАДИИ И КОНВЕЙЕРНЫЕ «ПУЗЫРЬКИ»

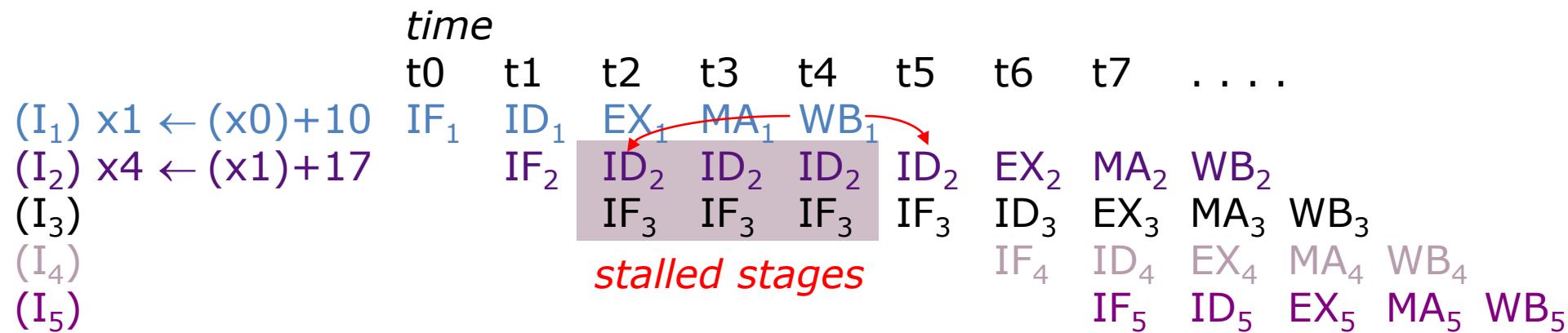


time

	t0	t1	t2	t3	t4	t5	t6	t7	...
IF	I_1	I_2	I_3	I_3	I_3	I_3	I_4	I_5	
ID		I_1	I_2	I_2	I_2	I_2	I_3	I_4	I_5
EX			I_1	-	-	-	I_2	I_3	I_4
MA				I_1	-	-	-	I_2	I_3
WB					I_1	-	-	-	I_2

Resource Usage

БЛОКИРОВАННЫЕ СТАДИИ И КОНВЕЙЕРНЫЕ «ПУЗЫРЬКИ»

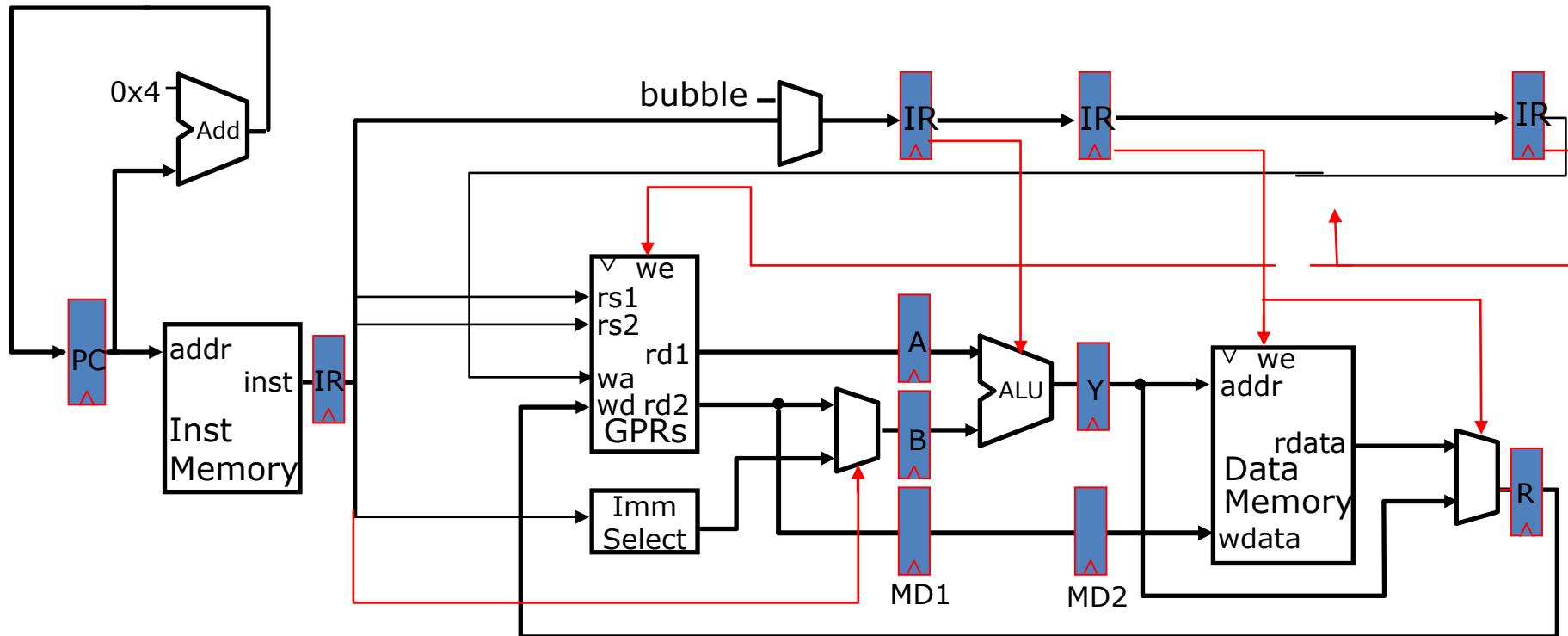


time

	t0	t1	t2	t3	t4	t5	t6	t7	...
IF	I_1	I_2	I_3	I_3	I_3	I_3	I_4	I_5	
ID		I_1	I_2	I_2	I_2	I_2	I_3	I_4	I_5
EX			I_1	-	-	-	I_2	I_3	I_4
MA				I_1	-	-	I_2	I_3	I_4
WB					I_1	-	-	I_2	I_3

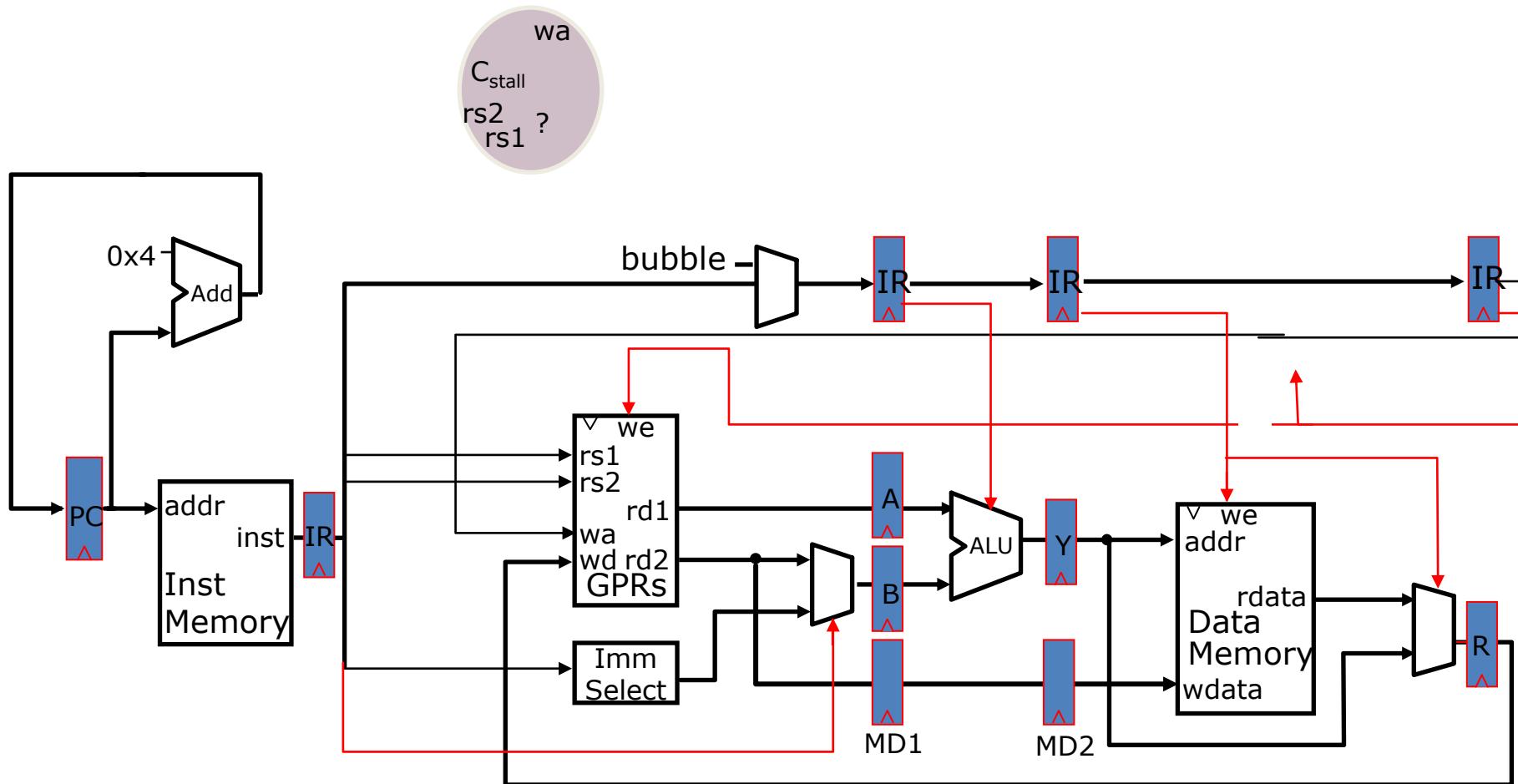
- \Rightarrow *pipeline bubble*

Interlock'и: управляющая логика



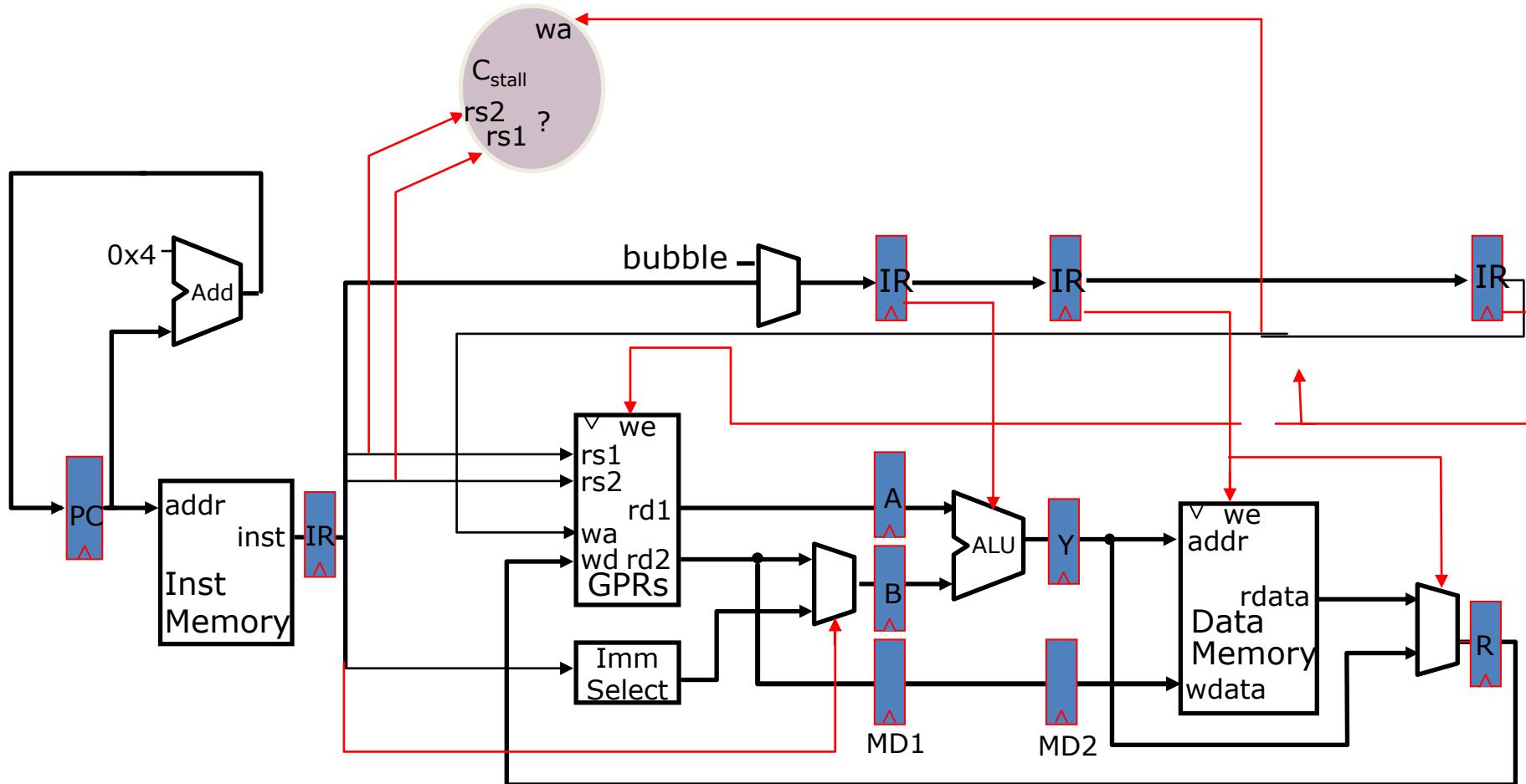
Сравниваются *регистры-источники (source registers)* на стадии decode с *регистрами назначения (destination registers)* у незавершенных инструкций

Interlock'и: управляющая логика



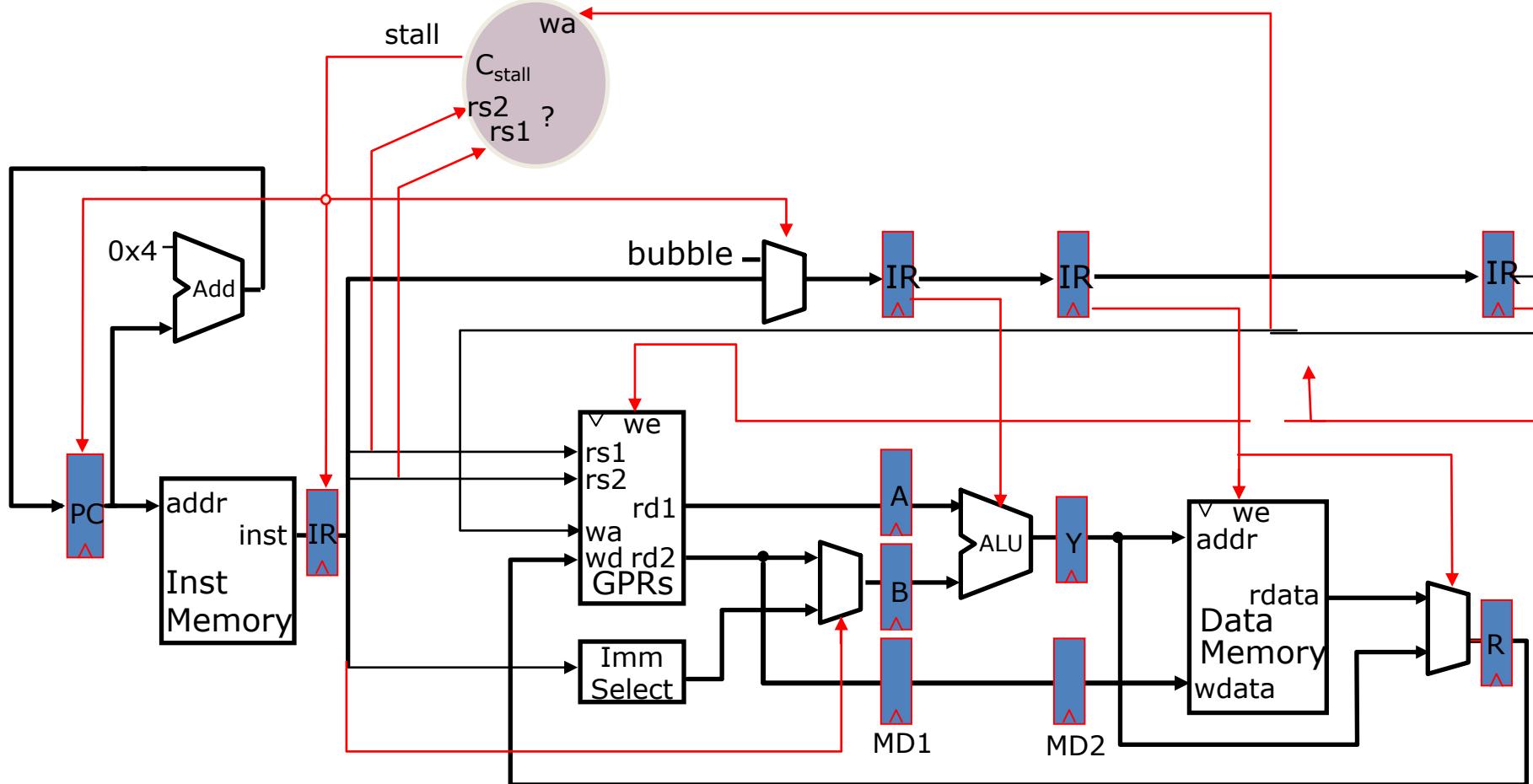
Сравниваются *регистры-источники (source registers)* на стадии decode с *регистрами назначения (destination registers)* у незавершенных инструкций

Interlock'и: управляющая логика



Сравниваются *регистры-источники (source registers)* на стадии decode с *регистрами назначения (destination registers)* у незавершенных инструкций

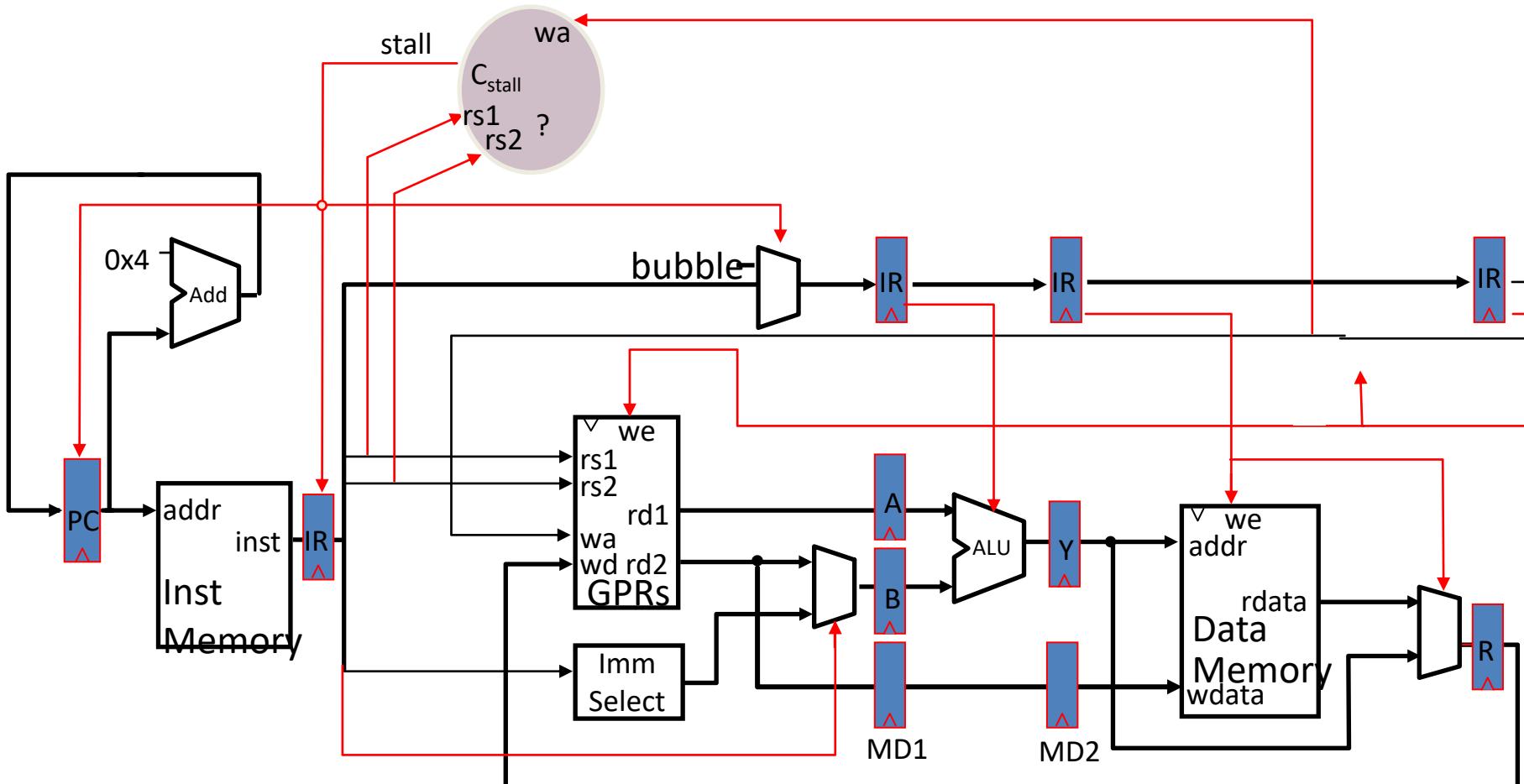
Interlock'и: управляющая логика



Сравниваются *регистры-источники (source registers)* на стадии decode с *регистрами назначения (destination registers)* у незавершенных инструкций

Interlock'и: управляющая логика

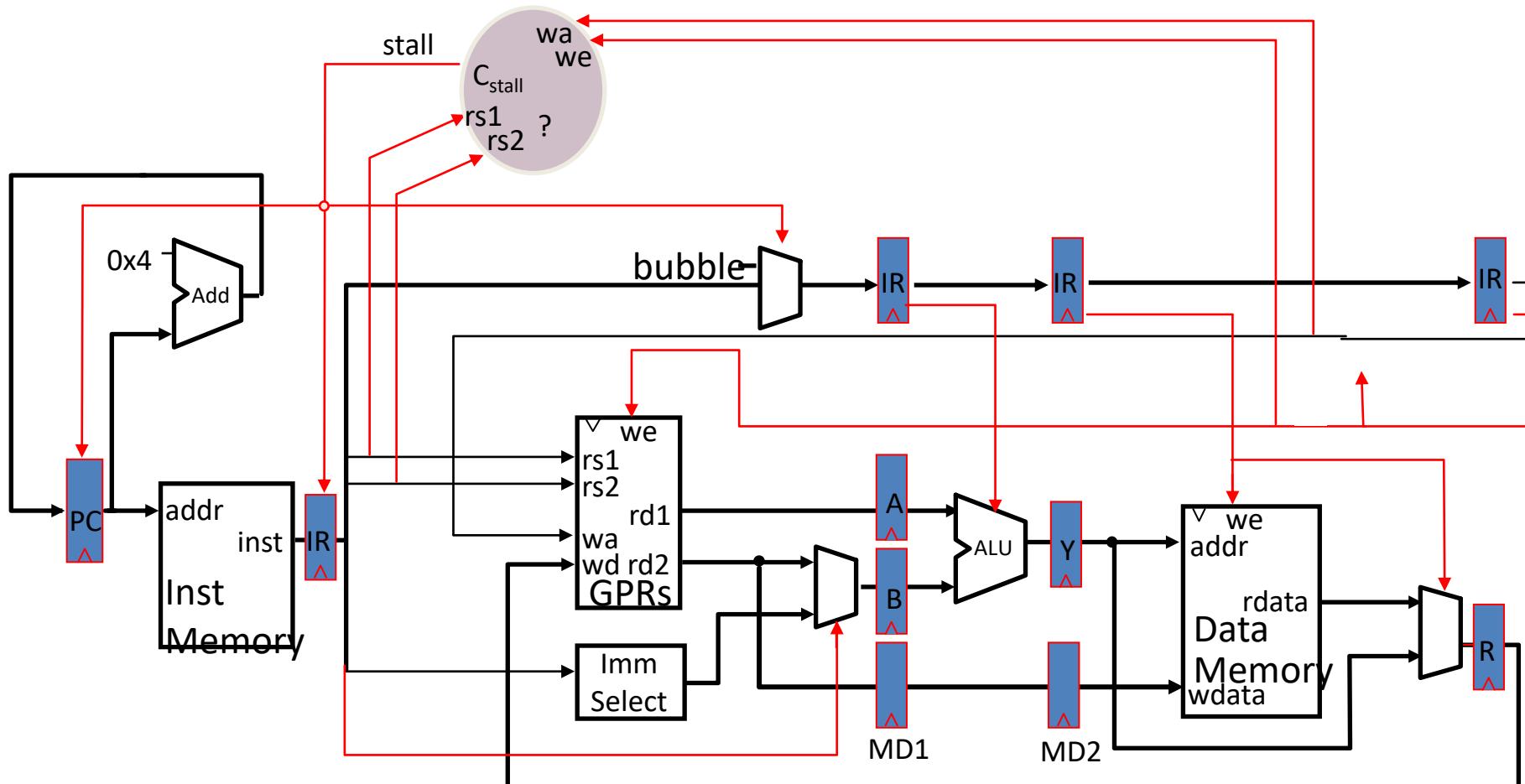
игнорирование условных и безусловных переходов



Требуется ли всегда делать stall, если поле rs совпадает с каким-нибудь rd?

Interlock'и: управляющая логика

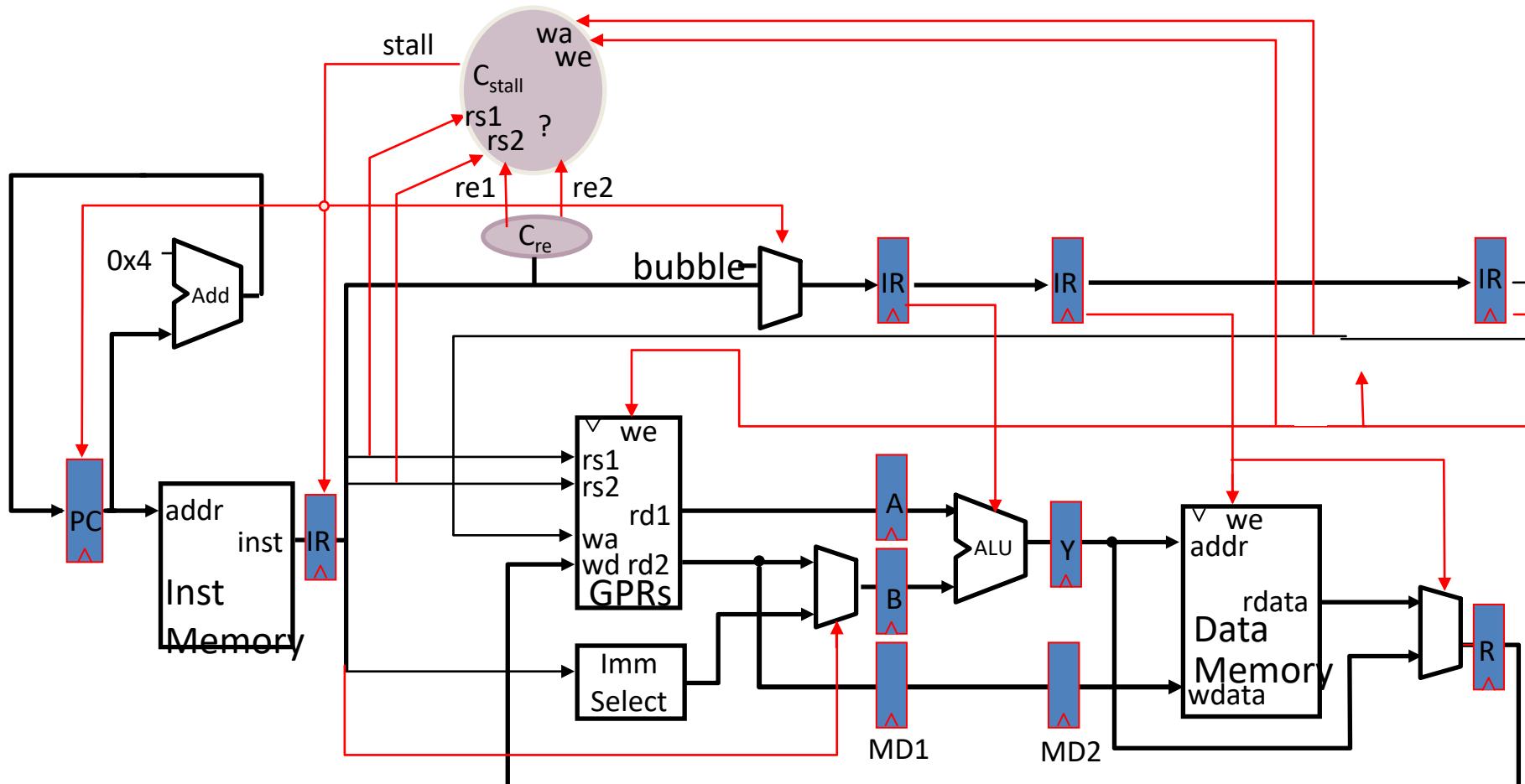
игнорирование условных и безусловных переходов



Требуется ли всегда делать stall, если поле rs совпадает с каким-нибудь rd?
Не каждая инструкция осуществляет запись в регистр $\Rightarrow we$

Interlock'и: управляющая логика

игнорирование условных и безусловных переходов



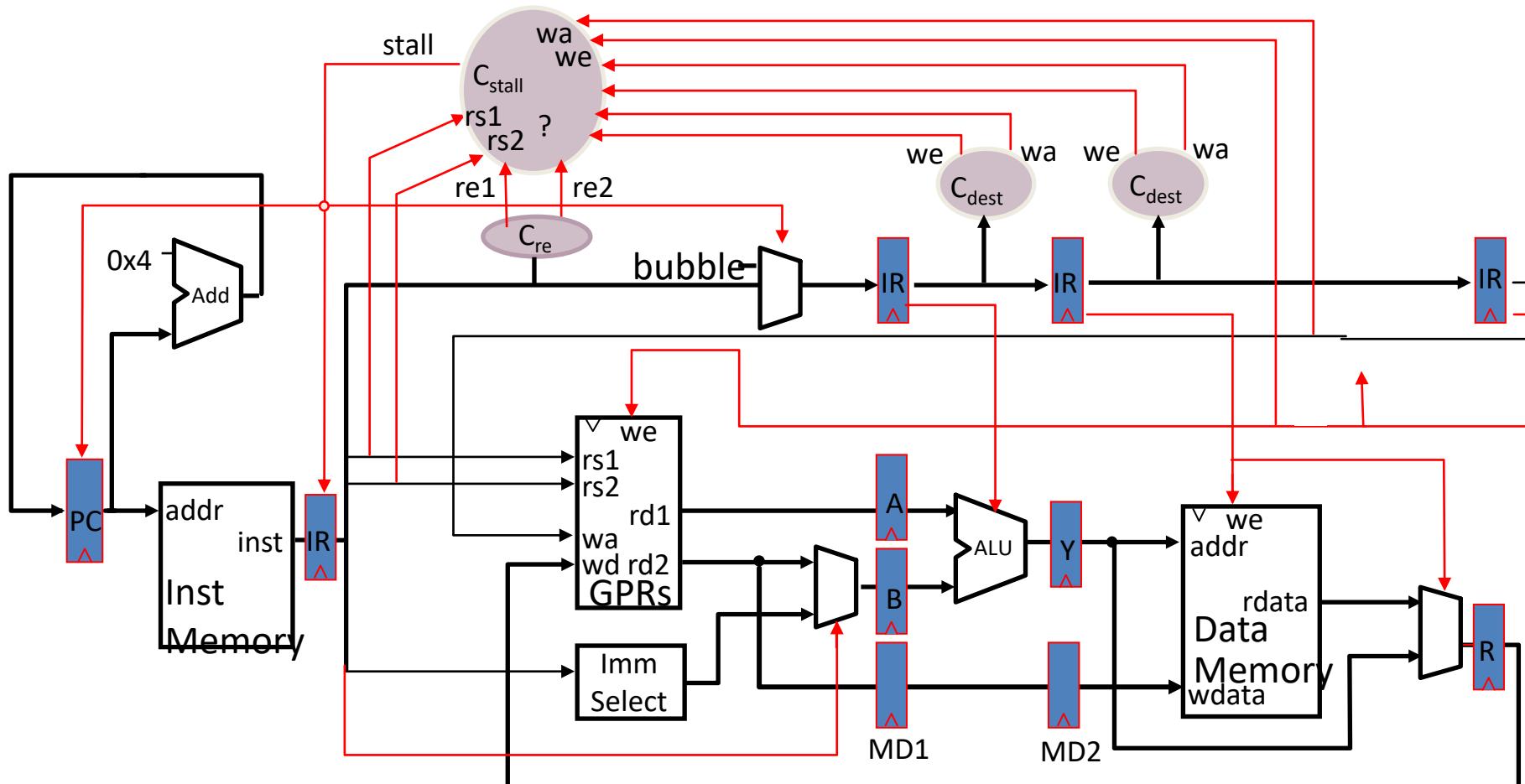
Требуется ли всегда делать stall, если поле rs совпадает с каким-нибудь rd?

Не каждая инструкция осуществляет запись в регистр => we

Не каждая инструкция осуществляет чтение из регистра => re

Interlock'и: управляющая логика

игнорирование условных и безусловных переходов



Требуется ли всегда делать stall, если поле rs совпадает с каким-нибудь rd?

Не каждая инструкция осуществляет запись в регистр => we

Не каждая инструкция осуществляет чтение из регистра => re

Регистры-источники и регистры назначения

	func7	rs2	rs1	func3	rd	opcode	ALU
	immediate12		rs1	func3	rd	opcode	ALUI/LW/JALR
	imm	rs2	rs1	func3	imm	opcode	SW/Bcond
	Jump Offset[19:0]				rd	opcode	
					$source^0(s)$	$destination$	
ALU	rd <= rs1	func10	rs2		rs1, rs2	rd	
ALUI	rd <= rs1	op imm			rs1	rd	
LW	rd <= M	[rs1 + imm]			rs1	rd	
SW	M [rs1 + imm] <=	rs2			rs1, rs2	-	
Bcond	rs1, rs2				rs1, rs2	-	
	<i>true:</i> PC <= PC + imm						
	<i>false:</i> PC <= PC + 4						
JAL	x1 <= PC	, PC <= PC + imm			-	rd	
JALR	rd <= PC	, PC <= rs1 + imm			rs1	rd	

Вычисление сигнала блокировки (stall)

C_{dest}

$ws = rd$

$we = Case$ opcode

ALU, ALUi, LW, JALR =>on

... =>off

Вычисление сигнала блокировки (stall)

C_{dest}

$ws = rd$

$we = Case$ opcode

ALU, ALUi, LW, JALR =>on

... =>off

C_{re}

$re1 = Case$ opcode

ALU, ALUi,

=>on

=>off

$re2 = Case$ opcode

=>on

->off

Вычисление сигнала блокировки (stall)

C_{dest}

$ws = rd$

$we = Case$ opcode

ALU, ALUi, LW, JALR =>on

... =>off

C_{re}

$re1 = Case$ opcode

ALU, ALUi,

LW, SW, Bcond,

JALR

JAL

=>on

=>off

$re2 = Case$ opcode

=>on

->off

Вычисление сигнала блокировки (stall)

C_{dest}

$ws = rd$

$we = Case$ opcode

ALU, ALUi, LW, JALR =>on

... =>off

C_{re}

$re1 = Case$ opcode

ALU, ALUi,

LW, SW, Bcond,

JALR

JAL

=>on

=>off

$re2 = Case$ opcode

ALU, SW, Bcond =>on

->off

Вычисление сигнала блокировки (stall)

C_{dest}

$ws = rd$

$we = Case$ opcode

ALU, ALUi, LW, JALR =>on

... =>off

C_{re}

$re1 = Case$ opcode

ALU, ALUi,

LW, SW, Bcond,

JALR

JAL

=>on

=>off

$re2 = Case$ opcode

ALU, SW, Bcond =>on

...

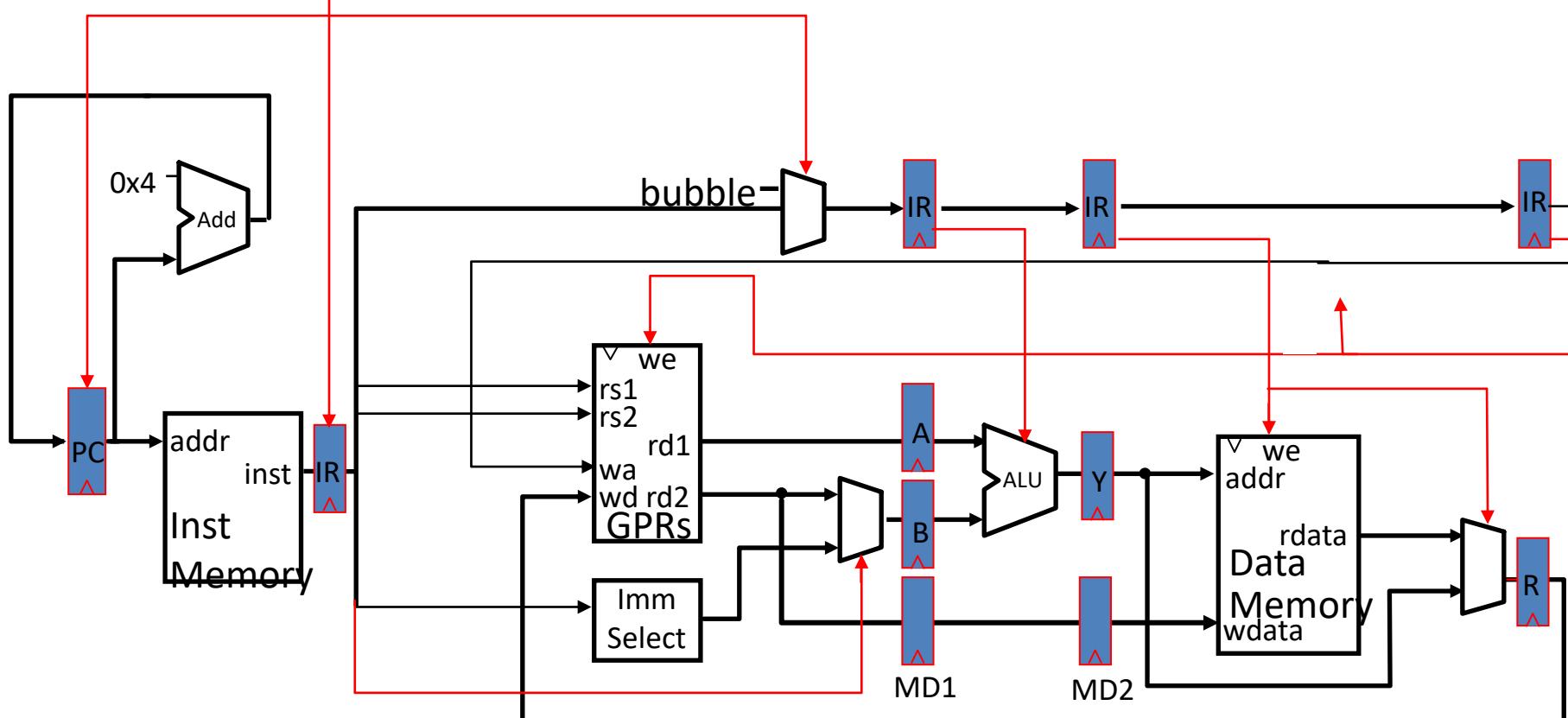
->off

C_{stall}

$$stall = ((rs1_D = ws_E).we_E + (rs1_D = ws_M).we_M + (rs1_D = ws_W).we_W) . re1_D + ((rs2_D = ws_E).we_E + (rs2_D = ws_M).we_M + (rs2_D = ws_W).we_W) . re2_D$$

Коллизии из-за Load'ов и Store'ов

Stall Condition



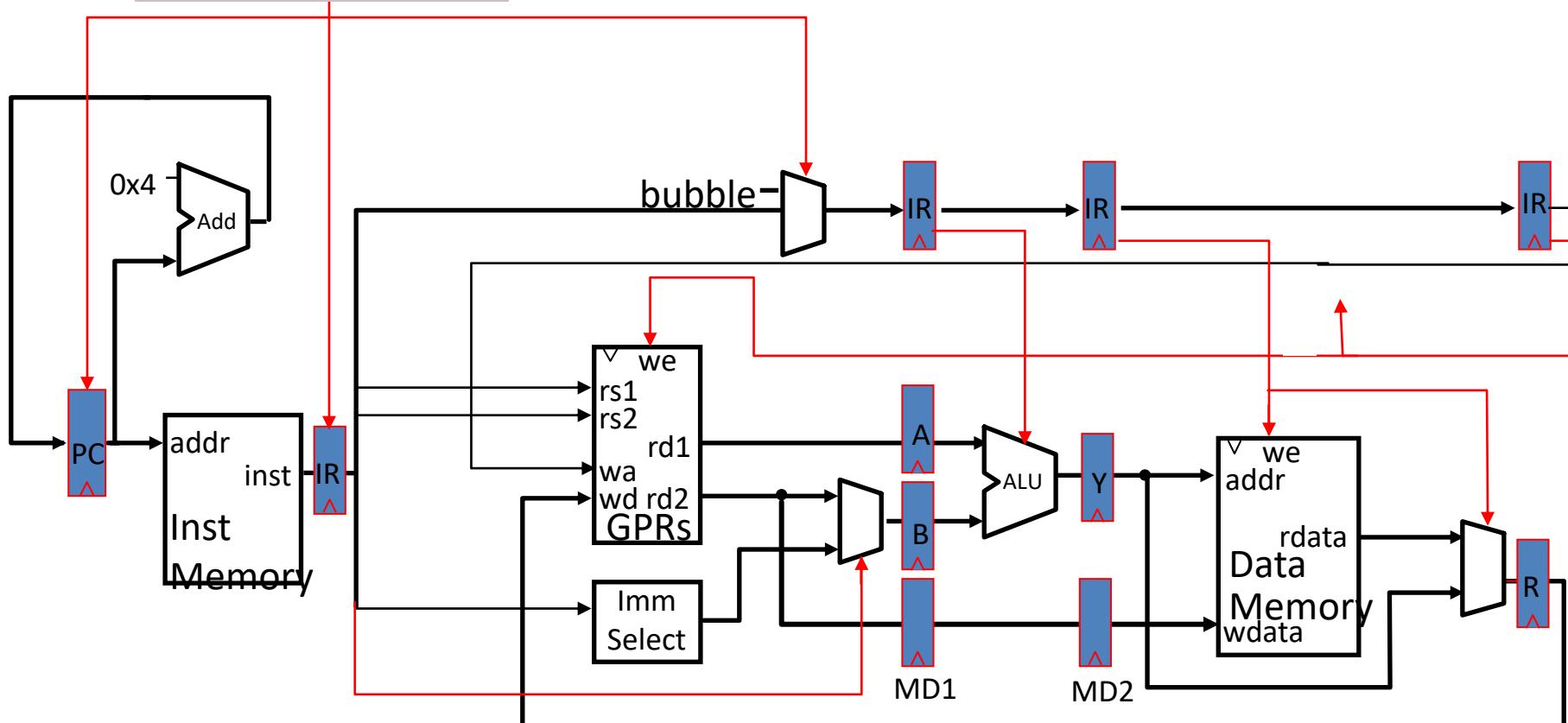
$$M[x_1+7] \leq x_2$$

$$x_4 \leq M[x_3+5]$$

...

Коллизии из-за Load'ов и Store'ов

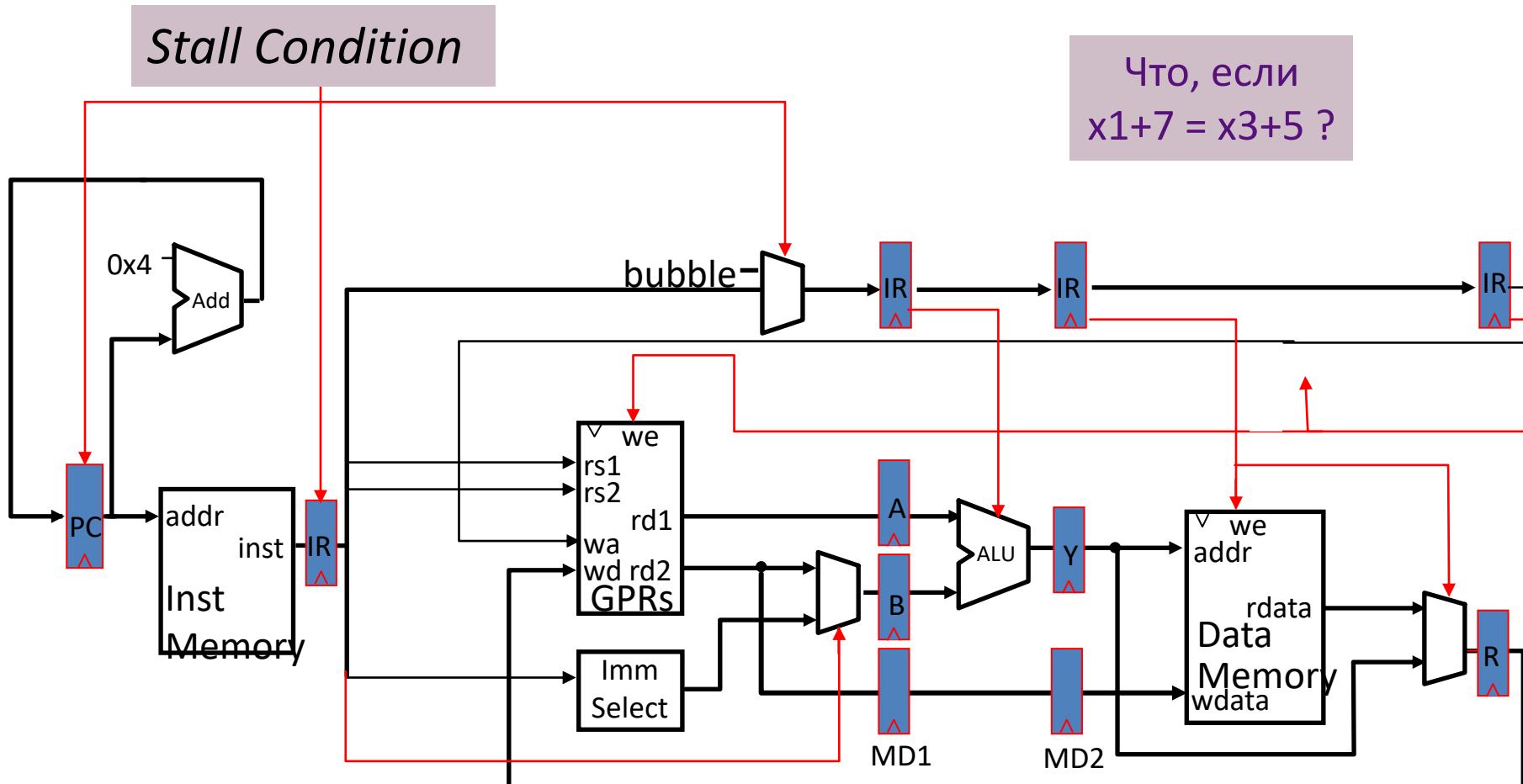
Stall Condition



...
 $M[x_1+7] \leq x_2$
 $x_4 \leq M[x_3+5]$
...

Возможна ли коллизия по данным в этой последовательности инструкций?

Коллизии из-за Load'ов и Store'ов



...
 $M[x_1+7] \leq x_2$
 $x_4 \leq M[x_3+5]$

Возможна ли коллизия по данным в этой последовательности инструкций?

Load- и Store-коллизии

...

$$M[x_1+7] \leq x_2$$
$$x_4 \leq M[x_3+5]$$

...

$x_1+7 = x_3+5 \Rightarrow \text{data hazard}$

Load- и Store-коллизии

```
...  
M[x1+7] <= x2  
x4 <= M[x3+5]  
...
```

$x_1+7 = x_3+5 \Rightarrow \text{data hazard}$

Однако коллизия избегается, т.к. в рассматриваемой системе записи происходят в 1 такт!

Load- и Store-коллизии

```
...  
M[x1+7] <= x2  
x4 <= M[x3+5]  
...
```

$x_1+7 = x_3+5 \Rightarrow \text{data hazard}$

Однако коллизия избегается, т.к. в рассматриваемой системе записи происходят в 1 такт!

Load-/Store-коллизии иногда разрешаются в конвейере или в самой системе памяти

Устранение зависимостей по данным (2)

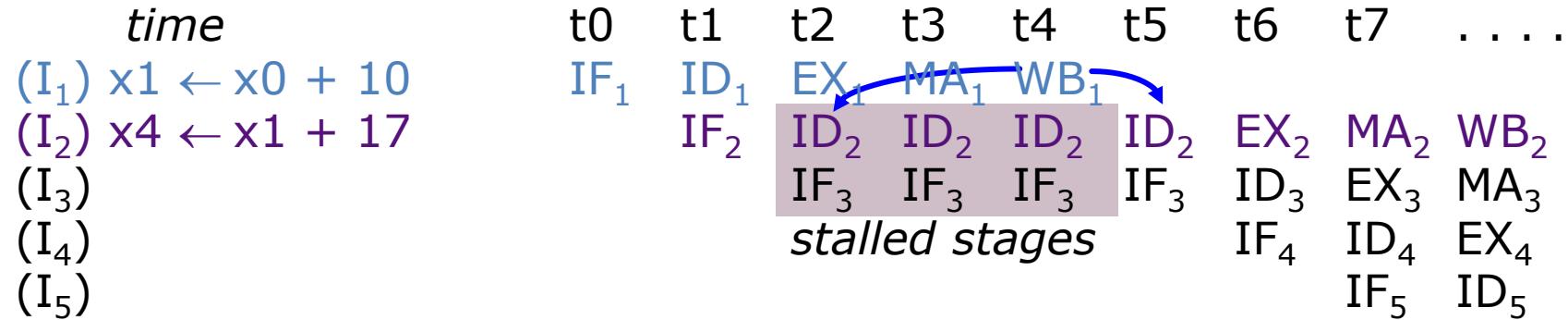
Стратегия 2:

*Сразу же, как они вычислены, перенаправлять данные на более ранние стадии конвейера → **байпас***

Байпасирование

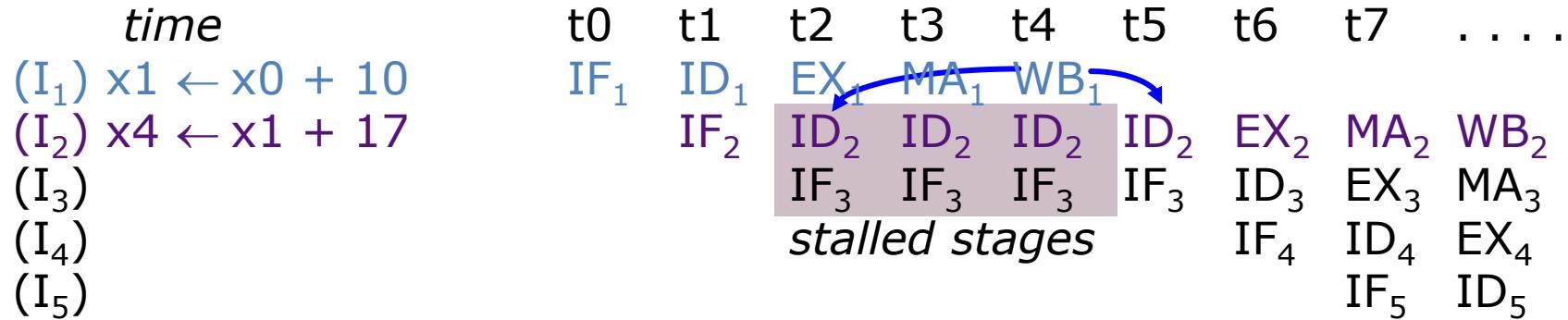
<i>time</i>	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) $x_1 \leftarrow x_0 + 10$	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) $x_4 \leftarrow x_1 + 17$	IF ₂	ID ₂	ID ₂	ID ₂	ID ₂	EX ₂	MA ₂	WB ₂	
(I ₃)		IF ₃	EX ₃	MA ₃					
(I ₄)						IF ₄	ID ₄	EX ₄	
(I ₅)						IF ₅	ID ₅		

Байпасирование



Каждая блокировка или *kill* добавляет «пузырек» (bubble) в конвейер
=> CPI > 1

Байпасирование



Каждая блокировка или *kill* добавляет «пузырек» (bubble) в конвейер
=> CPI > 1

Новый тракт данных, например, байпас, может передать данные с выхода ALU на его вход

Байпасирование

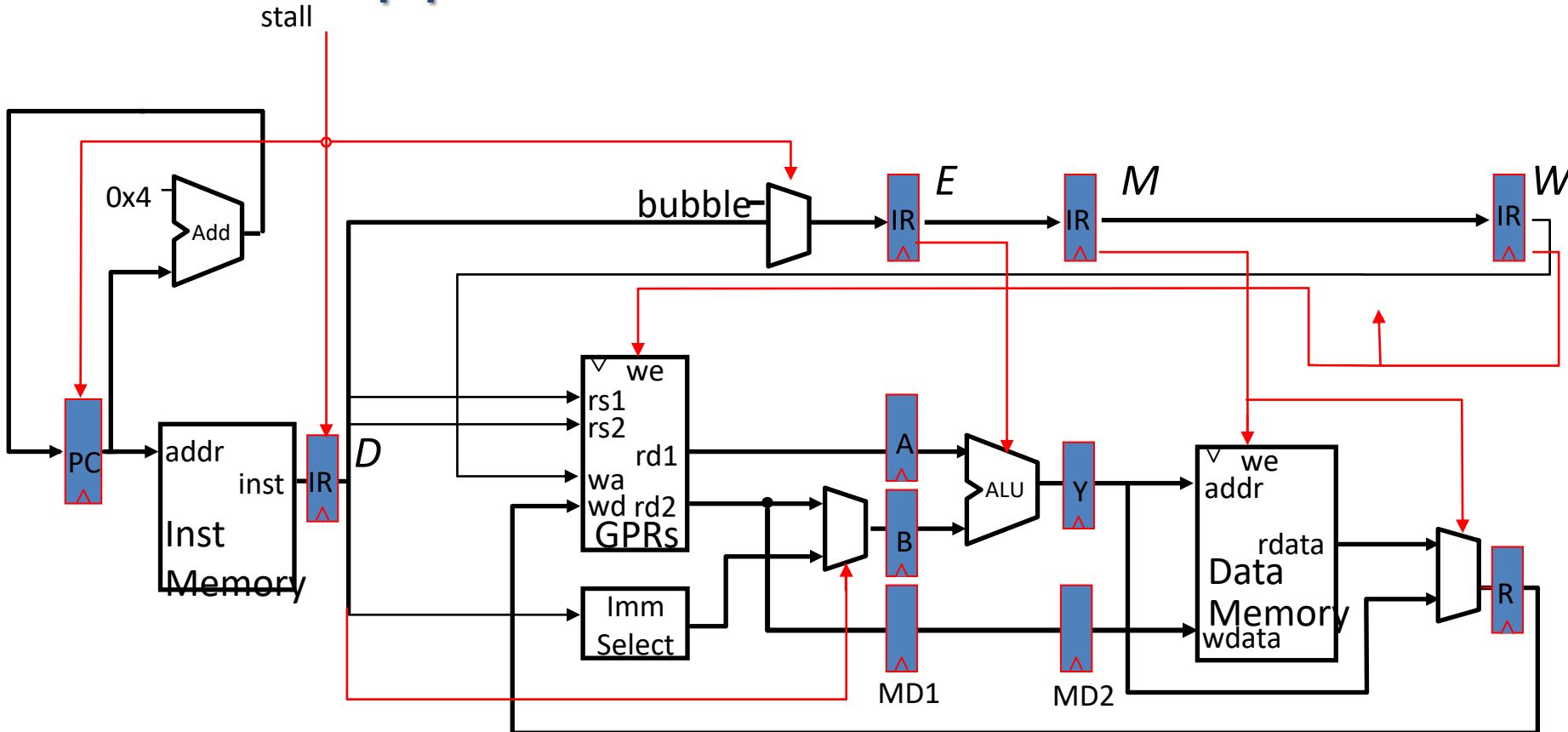
<i>time</i>	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) $x_1 \leftarrow x_0 + 10$	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) $x_4 \leftarrow x_1 + 17$		IF ₂	ID ₂	ID ₂	ID ₂	ID ₂	EX ₂	MA ₂	WB ₂
(I ₃)			IF ₃	EX ₃	MA ₃				
(I ₄)							IF ₄	ID ₄	EX ₄
(I ₅)							IF ₅	ID ₅	

Каждая блокировка или *kill* добавляет «пузырек» (bubble) в конвейер
=> CPI > 1

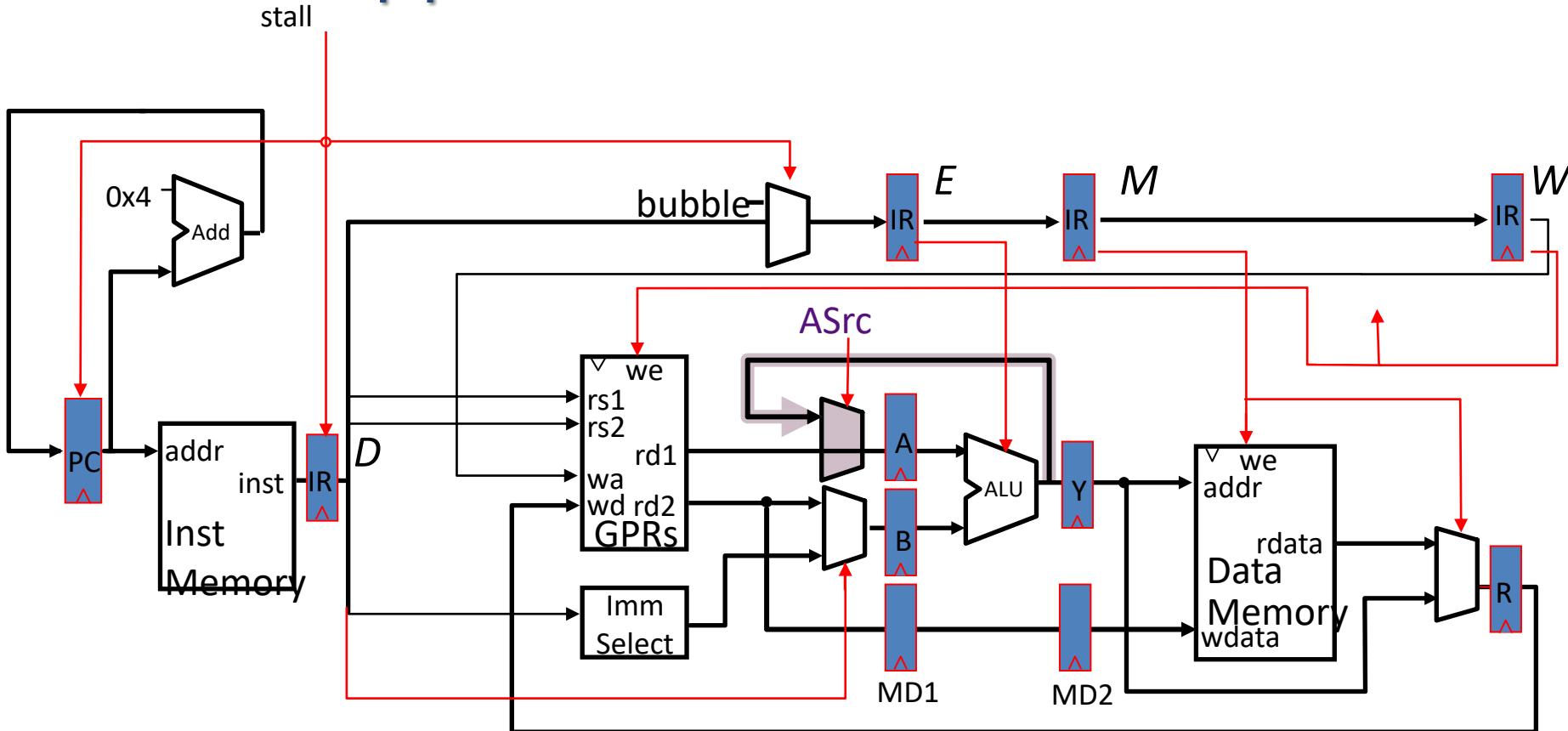
Новый тракт данных, например, байпас, может передать данные с выхода ALU на его вход

<i>time</i>	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) $x_1 \leftarrow x_0 + 10$	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) $x_4 \leftarrow x_1 + 17$		IF ₂	ID ₂	EX ₂	MA ₂	WB ₂			
(I ₃)			IF ₃	ID ₃	EX ₃	MA ₃	WB ₃		
(I ₄)				IF ₄	ID ₄	EX ₄	MA ₄	WB ₄	
(I ₅)					IF ₅	ID ₅	EX ₅	MA ₅	WB ₅

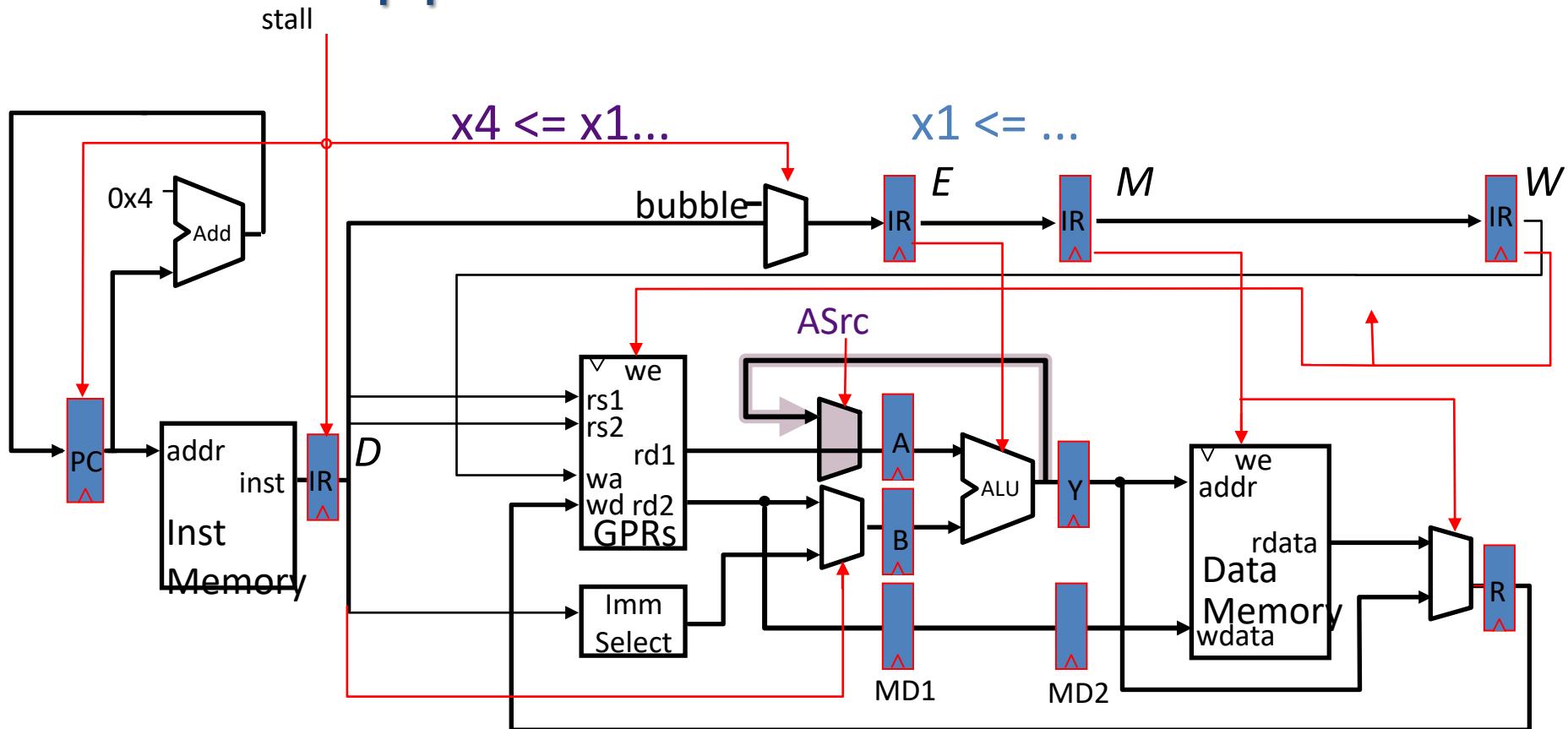
Добавление байпаса



Добавление байпаса



Добавление байпаса



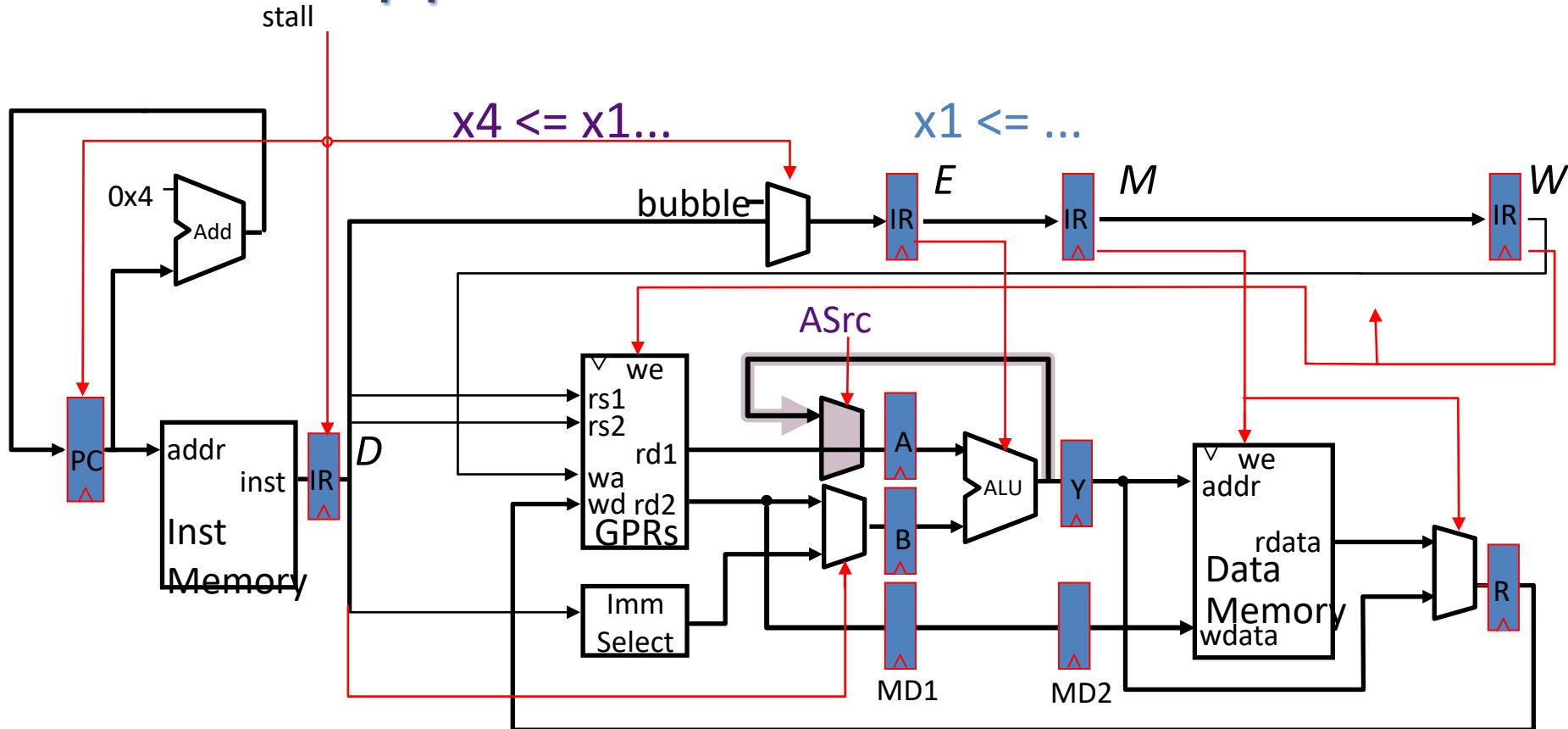
(I_1) ...
 (I_2)

$x1 \leq x0 + 10$
 $x4 \leq x1 + 17$

$x1 \leq M[x0 + 10]$
 $x4 \leq x1 + 17$

JAL 500
 $x4 \leq x1 + 17$

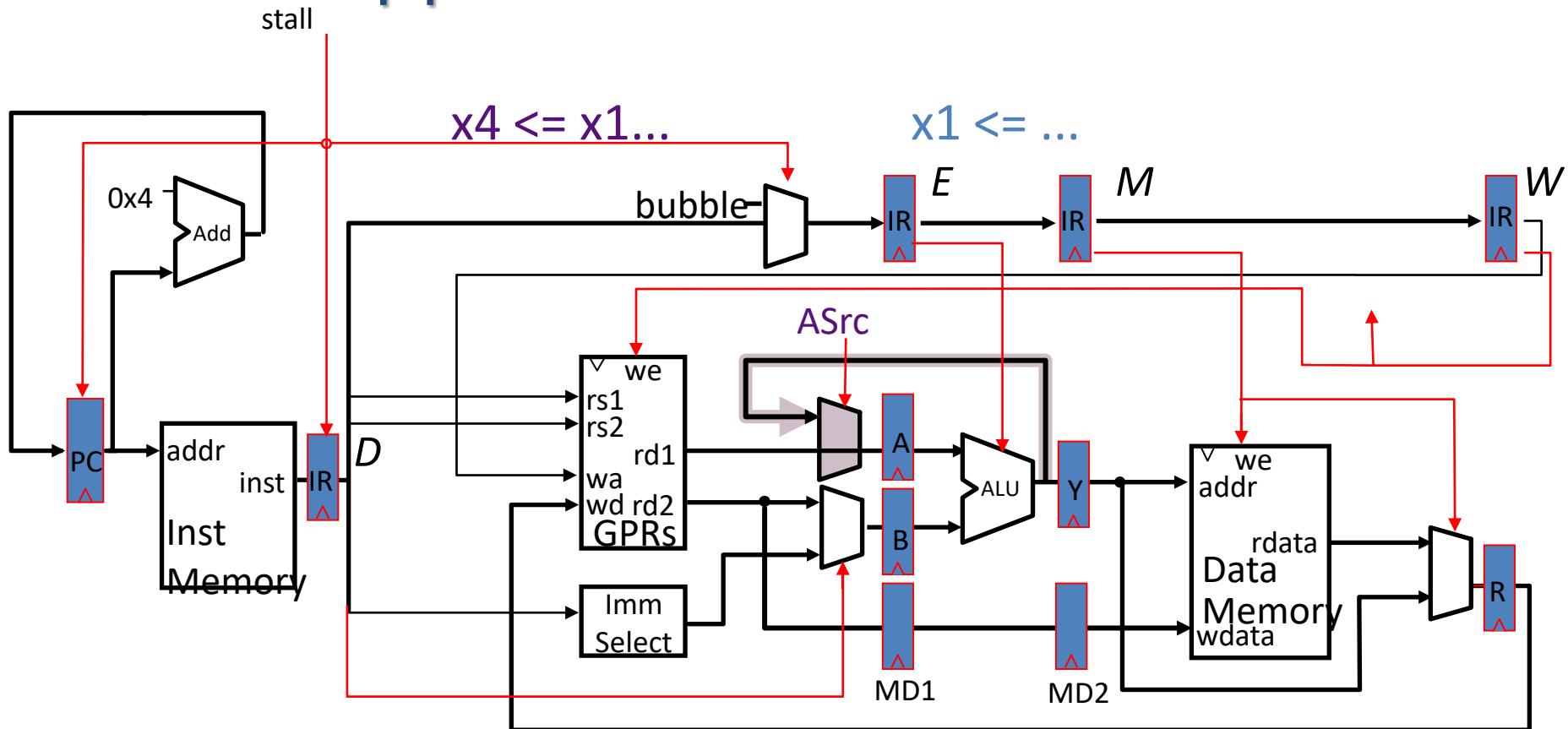
Добавление байпаса



Когда помогает этот байпас?

(I_1) (I_2)	\dots $x_1 \leq x_0 + 10$ $x_4 \leq x_1 + 17$	$x_1 \leq M[x_0 + 10]$ $x_4 \leq x_1 + 17$	$JAL\ 500$ $x_4 \leq x_1 + 17$
--------------------	---	---	-----------------------------------

Добавление байпаса



Когда помогает этот байпас?

$$\begin{array}{ll} (I_1) & \dots \\ (I_2) & x_1 \leq x_0 + 10 \\ & x_4 \leq x_1 + 17 \end{array}$$

yes

$$\begin{array}{l} x_1 \leq M[x_0 + 10] \\ x_4 \leq x_1 + 17 \end{array}$$

no

$$\begin{array}{l} JAL\ 500 \\ x_4 \leq x_1 + 17 \end{array}$$

no

Сигнал байпаса

Его вычисление из сигнала блокировки

```
stall = ( ((rs1D = wsE).weE + (rs1D = wsM).weM + (rs1D = wsW).weW).re1D
           +((rs2D = wsE).weE + (rs2D = wsM).weM + (rs2D = wsW).weW).re2D)
```

ws = rd

we = Case opcode
ALU, ALUi, LW,, JAL JALR => on
... => off

Сигнал байпаса

Его вычисление из сигнала блокировки

```
stall = ( ((rs1D = wsE).weE + (rs1D = wsM).weM + (rs1D = wsW).weW).re1D
          +((rs2D = wsE).weE + (rs2D = wsM).weM + (rs2D = wsW).weW).re2D)
```

ws = rd

we = Case opcode

ALU, ALUi, LW,, JAL JALR => on
... => off

Сигнал байпаса

Его вычисление из сигнала блокировки

$$\text{stall} = ((\text{rs1}_D = \text{ws}_E) \cdot \text{we}_E + (\text{rs1}_D = \text{ws}_M) \cdot \text{we}_M + (\text{rs1}_D = \text{ws}_W) \cdot \text{we}_W) \cdot \text{re1}_D \\ + ((\text{rs2}_D = \text{ws}_E) \cdot \text{we}_E + (\text{rs2}_D = \text{ws}_M) \cdot \text{we}_M + (\text{rs2}_D = \text{ws}_W) \cdot \text{we}_W) \cdot \text{re2}_D$$

$\text{ws} = \text{rd}$

$\text{we} = \text{Case opcode}$

ALU, ALUi, LW,, JAL JALR => on
... => off

$$\text{ASrc} = (\text{rs1}_D = \text{ws}_E) \cdot \text{we}_E \cdot \text{re1}_D$$

Сигнал байпаса

Его вычисление из сигнала блокировки

$$\text{stall} = ((\text{rs1}_D = \text{ws}_E) \cdot \text{we}_E + (\text{rs1}_D = \text{ws}_M) \cdot \text{we}_M + (\text{rs1}_D = \text{ws}_W) \cdot \text{we}_W) \cdot \text{re1}_D \\ + ((\text{rs2}_D = \text{ws}_E) \cdot \text{we}_E + (\text{rs2}_D = \text{ws}_M) \cdot \text{we}_M + (\text{rs2}_D = \text{ws}_W) \cdot \text{we}_W) \cdot \text{re2}_D$$

$\text{ws} = \text{rd}$

$\text{we} = \text{Case opcode}$

ALU, ALUi, LW,, JAL JALR => on
... => off

$$\text{ASrc} = (\text{rs1}_D = \text{ws}_E) \cdot \text{we}_E \cdot \text{re1}_D$$

Верно ли это?

Сигнал байпаса

Его вычисление из сигнала блокировки

$$\text{stall} = ((\text{rs1}_D = \text{ws}_E) \cdot \text{we}_E + (\text{rs1}_D = \text{ws}_M) \cdot \text{we}_M + (\text{rs1}_D = \text{ws}_W) \cdot \text{we}_W) \cdot \text{re1}_D \\ + ((\text{rs2}_D = \text{ws}_E) \cdot \text{we}_E + (\text{rs2}_D = \text{ws}_M) \cdot \text{we}_M + (\text{rs2}_D = \text{ws}_W) \cdot \text{we}_W) \cdot \text{re2}_D$$

$\text{ws} = \text{rd}$

$\text{we} = \text{Case opcode}$
ALU, ALUi, LW, JAL JALR => on
... => off

$$\text{ASrc} = (\text{rs1}_D = \text{ws}_E) \cdot \text{we}_E \cdot \text{re1}_D$$

Верно ли это?

Нет, потому что этот байпас полезен только для ALU- и ALUi-инструкций

Сигнал байпаса

Его вычисление из сигнала блокировки

$$\text{stall} = ((\text{rs1}_D = \text{ws}_E) \cdot \text{we}_E + (\text{rs1}_D = \text{ws}_M) \cdot \text{we}_M + (\text{rs1}_D = \text{ws}_W) \cdot \text{we}_W) \cdot \text{re1}_D \\ + ((\text{rs2}_D = \text{ws}_E) \cdot \text{we}_E + (\text{rs2}_D = \text{ws}_M) \cdot \text{we}_M + (\text{rs2}_D = \text{ws}_W) \cdot \text{we}_W) \cdot \text{re2}_D$$

$\text{ws} = \text{rd}$

$\text{we} = \text{Case opcode}$
ALU, ALUi, LW, JAL JALR => on
... => off

$$\text{ASrc} = (\text{rs1}_D = \text{ws}_E) \cdot \text{we}_E \cdot \text{re1}_D$$

Верно ли это?

Нет, потому что этот байпас полезен только для ALU- и ALUi-инструкций

Можно разделить we_E на две компоненты: we-bypass и we-stall

Сигналы байпаса и блокировки

Разделим we_E на две компоненты: we-bypass и we-stall

$we\text{-bypass}_E = \text{Case } opcode_E$

ALU, ALUi => on

... => off

$we\text{-stall}_E = \text{Case } opcode_E$

LW, JAL, JALR=> on

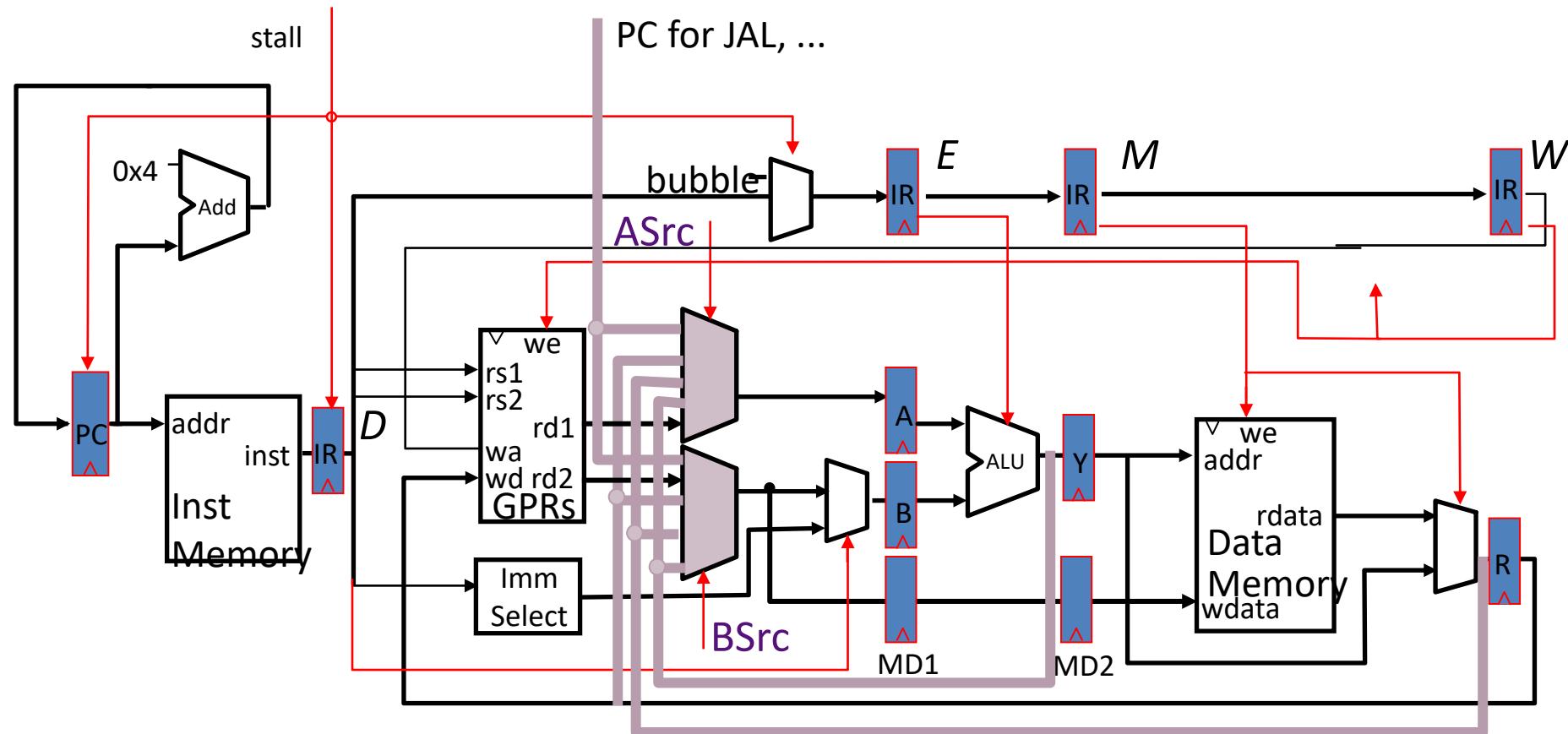
JAL => on

... => off

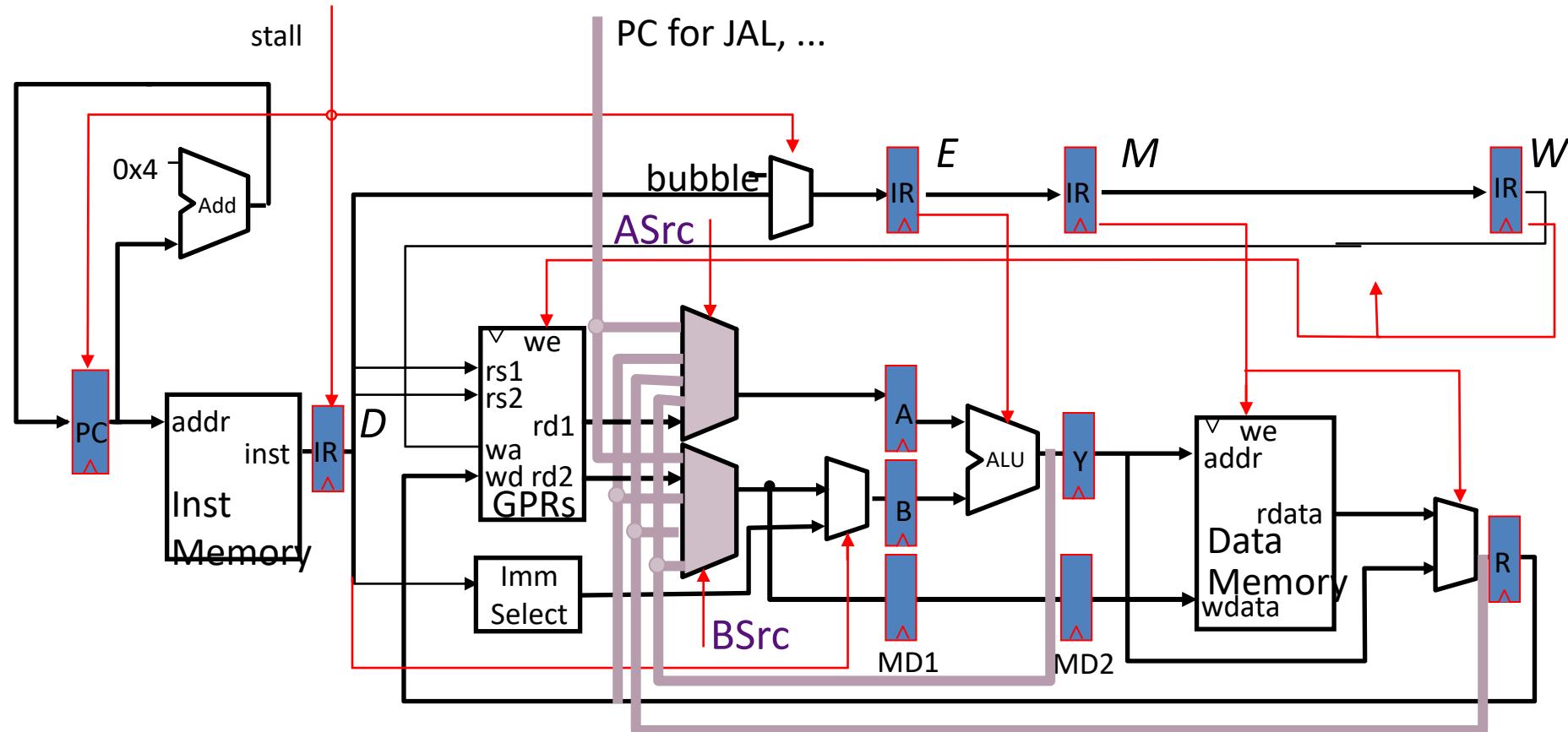
$ASrc = (rs1_D = ws_E).we\text{-bypass}_E . re1_D$

$stall = ((rs1_D = ws_E).we\text{-stall}_E +$
 $(rs1_D = ws_M).we_M + (rs1_D = ws_W).we_W) . re1_D$
 $+ (rs2_D = ws_E).we_E + (rs2_D = ws_M).we_M + (rs2_D = ws_W).we_W) . re2_D$

Полностью байпасированный тракт данных

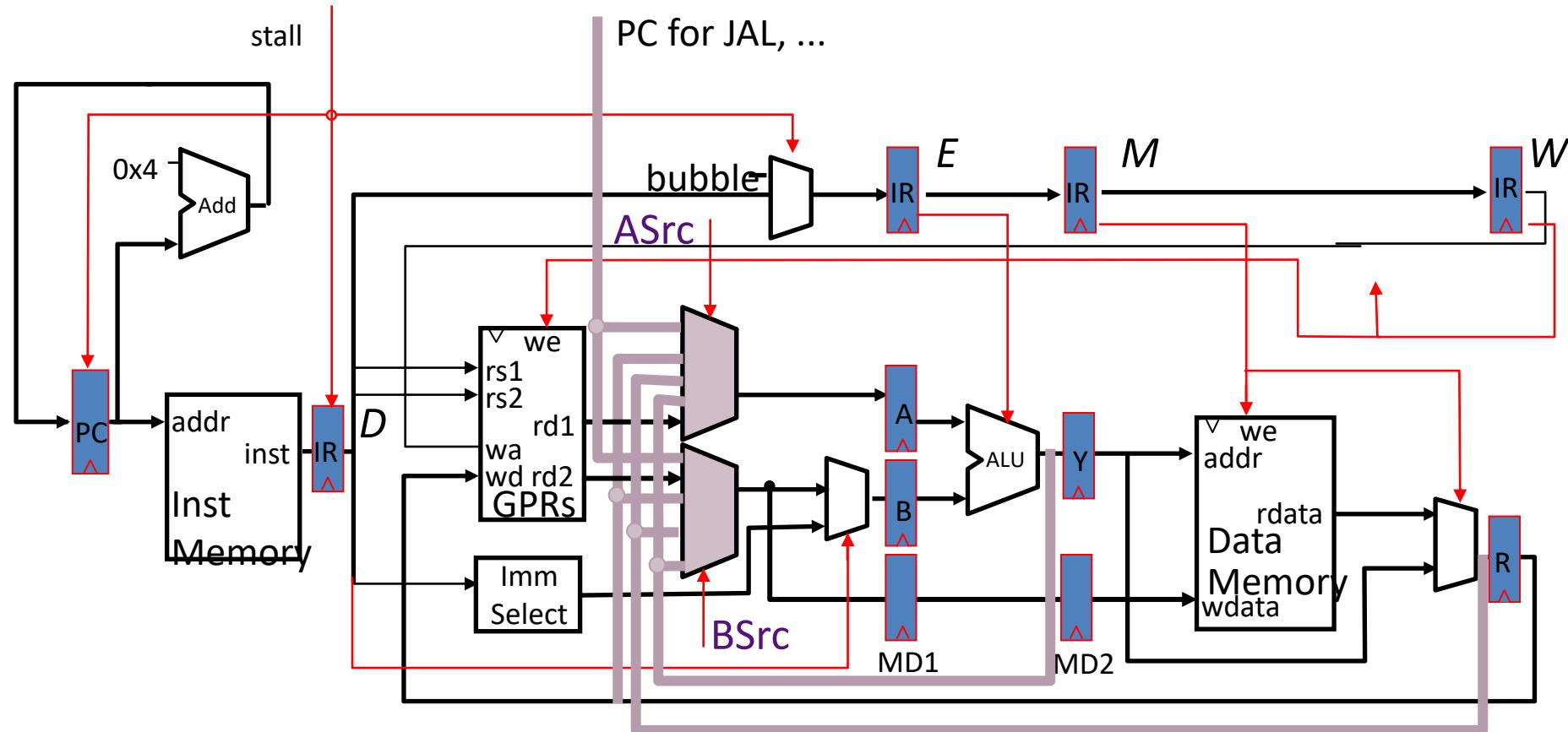


Полностью байпасированный тракт данных



Нужен ли все еще
сигнал блокировки?

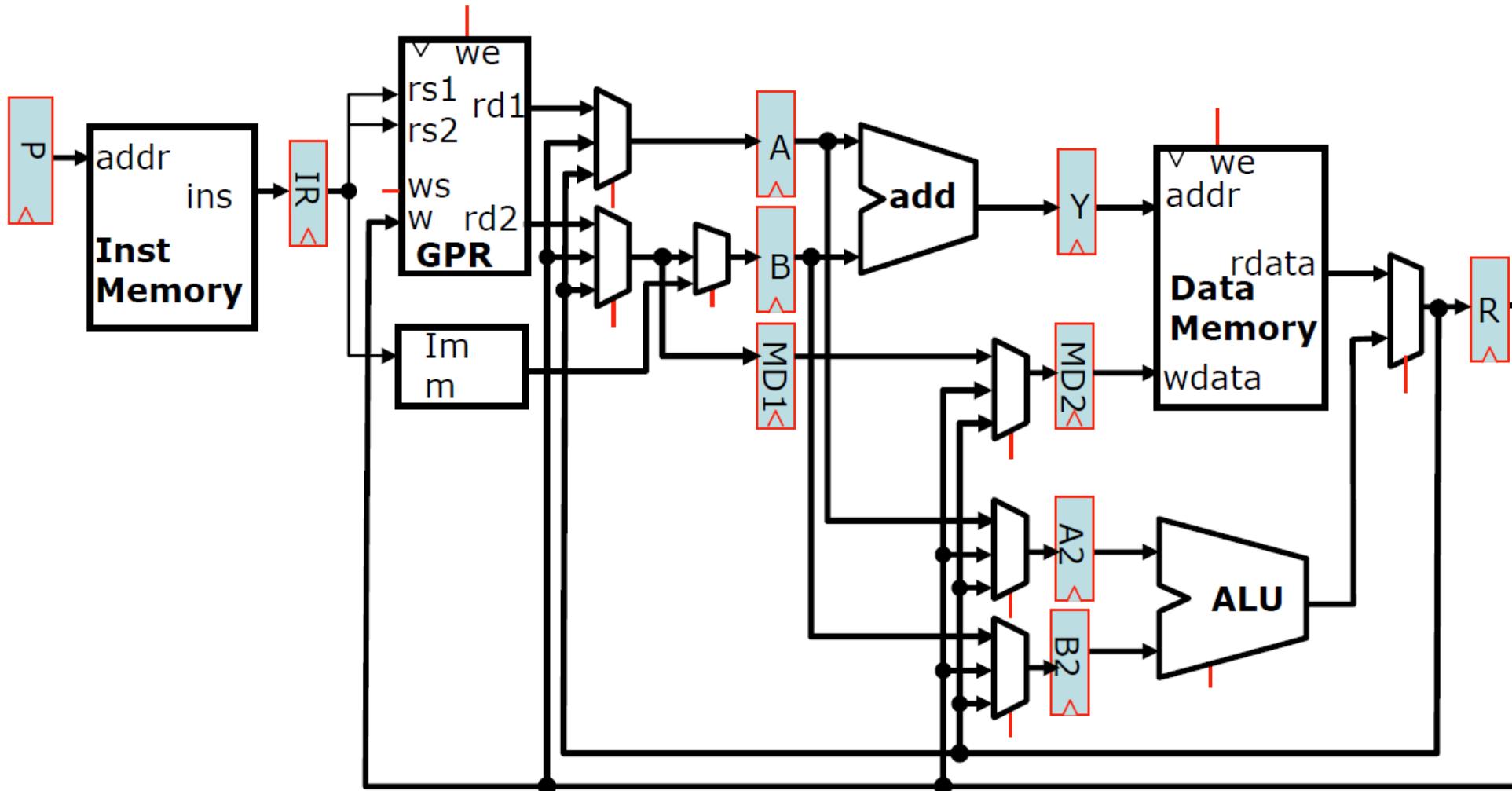
Полностью байпасированный тракт данных



Нужен ли все еще
сигнал блокировки?

$$\begin{aligned} \text{stall} = & (rs1_D = ws_E) \cdot (\text{opcode}_E = LW_E) \cdot (ws_E \neq 0) \cdot re1_D \\ & + (rs2_D = ws_E) \cdot (\text{opcode}_E = LW_E) \cdot (ws_E \neq 0) \cdot re2_D \end{aligned}$$

Добавление сумматора



Изменения

- Новые стадии

IF -> ID -> AC -> EX/MEM -> WB

- Разрешение коллизии

$(r2) \leftarrow M[(r1) + 7]$

$(r4) \leftarrow (r2) + 3$

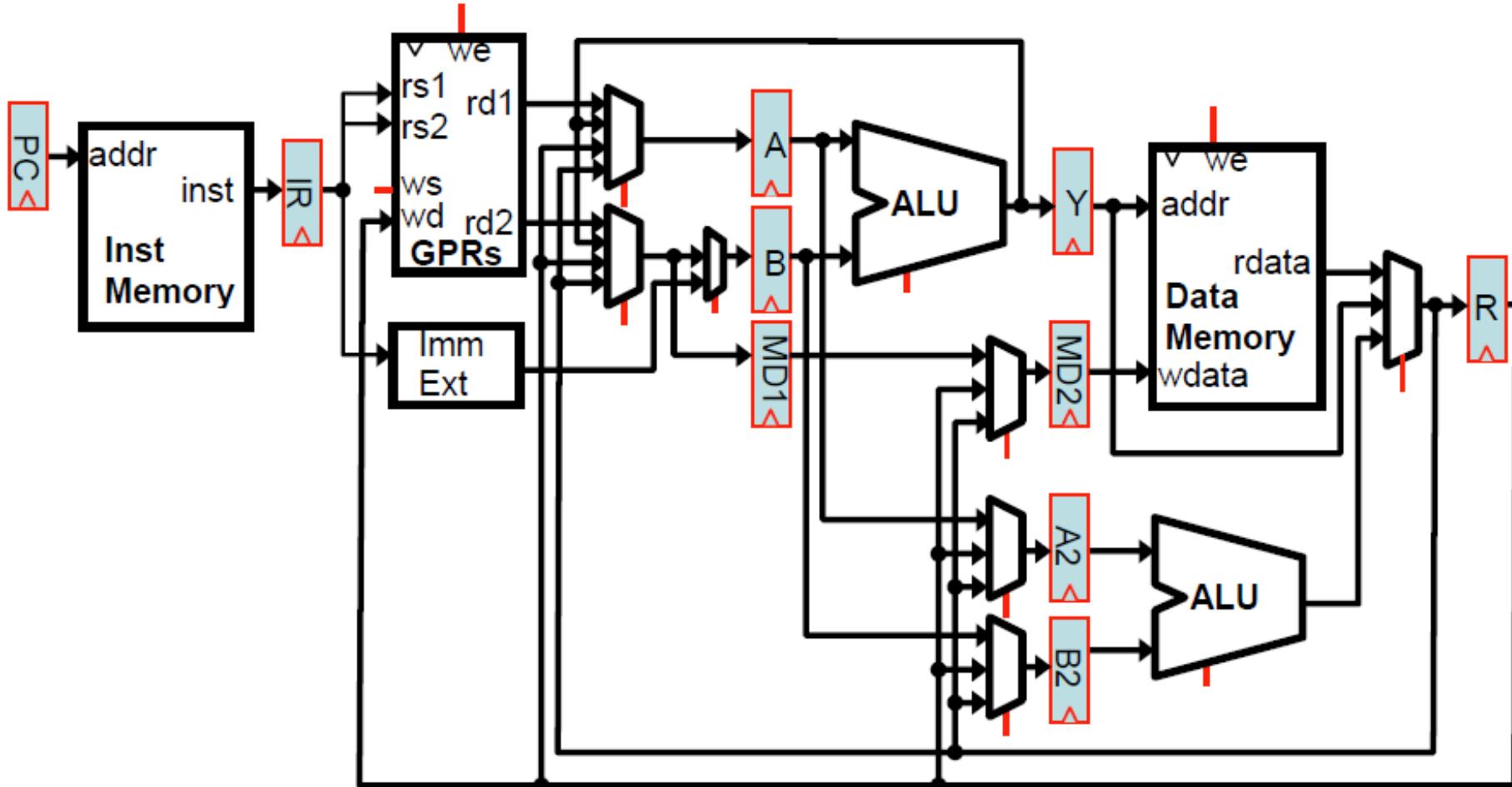
Недостатки

- Новая коллизия:

$$(r2) \leftarrow (r1) + 6$$
$$M[(r2) + 14] \leftarrow (r3)$$

Как её устраниТЬ?

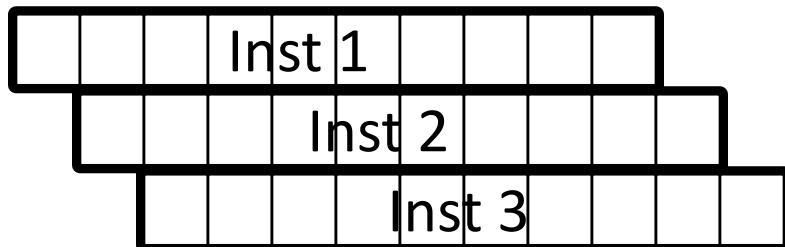
Добавление ALU



IF -> ID -> EX1 -> EX2/MEM -> WB

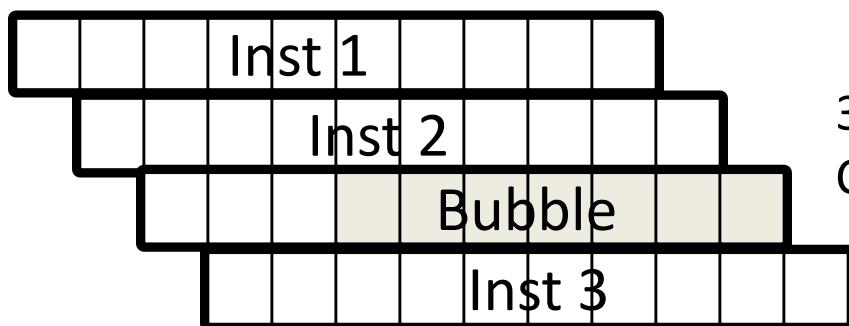
Примеры CPI в конвейере

Time →

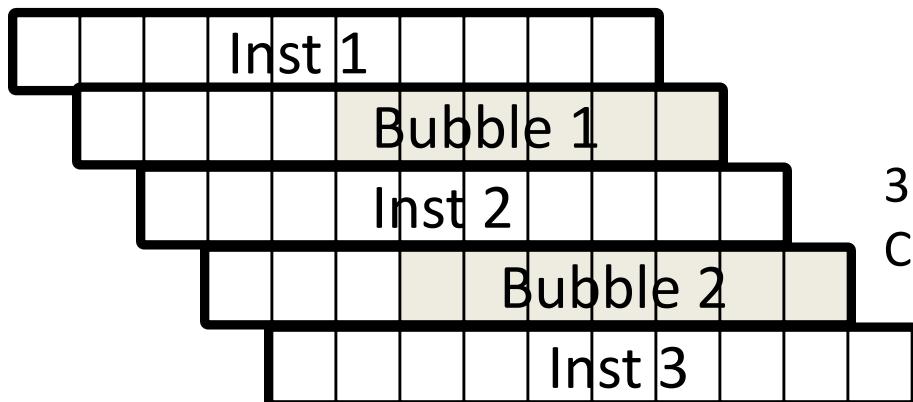


Измерение от окончания исполнения первой инструкции до окончания исполнения последней инструкции в последовательности

3 instructions finish in 3 cycles
 $CPI = 3/3 = 1$



3 instructions finish in 4 cycles
 $CPI = 4/3 = 1.33$



3 instructions finish in 5cycles
 $CPI = 5/3 = 1.67$

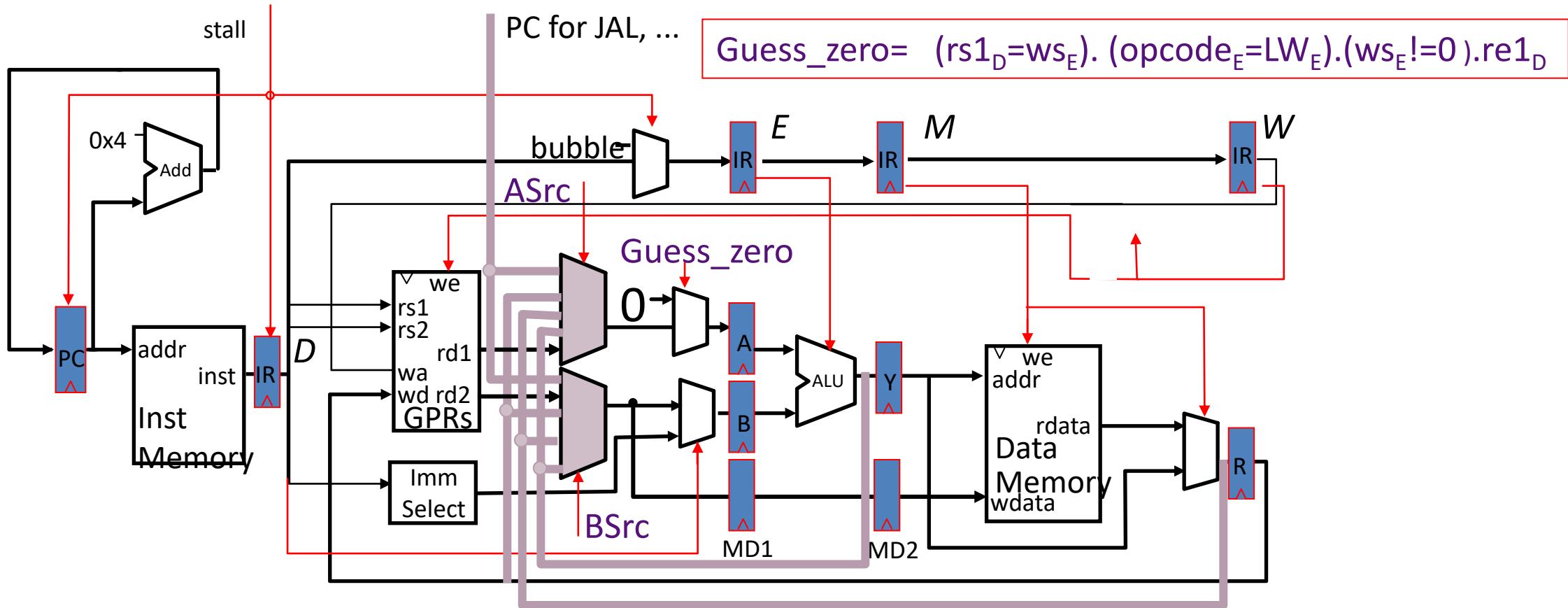
Устранение коллизий по данным (3)

Стратегия 3: Спекулирование о зависимости!

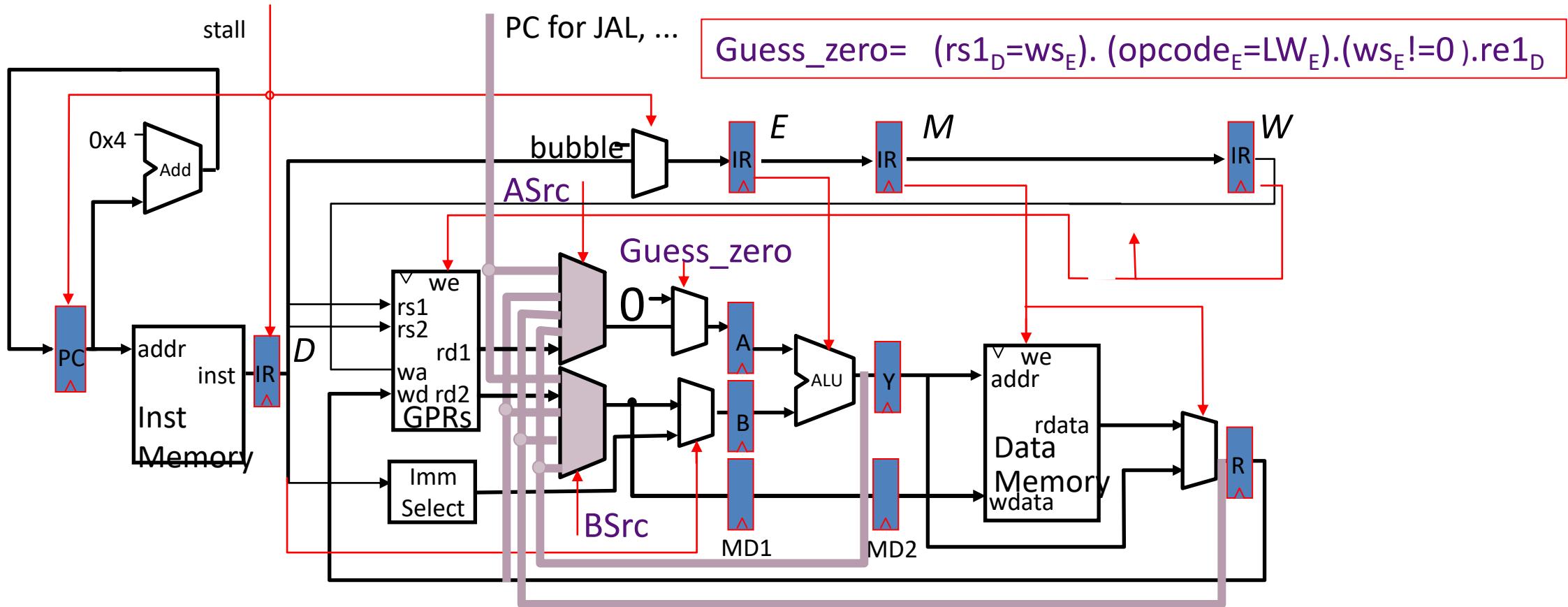
Два случая:

- *Угадали:* ничего не делать
- *Не угадали:* «убить» и начать заново

Спекуляция о том, что load value = 0



Спекуляция о том, что load value = 0



Помимо этого, нужно добавить аппаратуру, чтобы помнить о том, что это было предположение, а также делать flush конвейера, если оно было неверным!

Слишком затратно, чтобы делать на практике

Коллизии по управлению

Что нужно сделать, чтобы вычислить следующее значение РС?

Для команд Jump:

- Код операции, РС и offset

Для команд Jump Register:

- Код операции, значение регистра и РС

Для команд условного ветвления (Branch Condition):

- Код операции, регистр (для условия), РС и offset

Для прочих команд:

- Код операции и РС (нужно знать, что команда не принадлежит ни к одному из типов выше)

PC Calculation Bubbles

time

t0 t1 t2 t3 t4 t5 t6 t7 . . .

PC Calculation Bubbles

	<i>time</i>								...
	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) x1 ← x0 + 10	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				

PC Calculation Bubbles

	<i>time</i>								...
	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) x1 ← x0 + 10	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) x3 ← x2 + 17		IF ₂	IF ₂	ID ₂	EX ₂	MA ₂	WB ₂		

PC Calculation Bubbles

	time								
	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) $x_1 \leftarrow x_0 + 10$	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) $x_3 \leftarrow x_2 + 17$		IF ₂	IF ₂	ID ₂	EX ₂	MA ₂	WB ₂		
(I ₃)				IF ₃	IF ₃	ID ₃	EX ₃	MA ₃	WB ₃

PC Calculation Bubbles

	<i>time</i>									
	t0	t1	t2	t3	t4	t5	t6	t7	...	
(I ₁) $x1 \leftarrow x0 + 10$	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁					
(I ₂) $x3 \leftarrow x2 + 17$		IF ₂	IF ₂	ID ₂	EX ₂	MA ₂	WB ₂			
(I ₃)				IF ₃	ID ₃	EX ₃	MA ₃	WB ₃		
(I ₄)					IF ₄	IF ₄	ID ₄	EX ₄	MA ₄	WB ₄

PC Calculation Bubbles

	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) $x1 \leftarrow x0 + 10$	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) $x3 \leftarrow x2 + 17$		IF ₂	IF ₂	ID ₂	EX ₂	MA ₂	WB ₂		
(I ₃)			IF ₃	IF ₃	ID ₃	EX ₃	MA ₃	WB ₃	
(I ₄)				IF ₄	IF ₄	ID ₄	EX ₄	MA ₄	WB ₄

PC Calculation Bubbles

	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) $x_1 \leftarrow x_0 + 10$	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) $x_3 \leftarrow x_2 + 17$		IF ₂	IF ₂	ID ₂	EX ₂	MA ₂	WB ₂		
(I ₃)			IF ₃	IF ₃	ID ₃	EX ₃	MA ₃	WB ₃	
(I ₄)				IF ₄	IF ₄	ID ₄	EX ₄	MA ₄	WB ₄

	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7	...

*Resource
Usage*

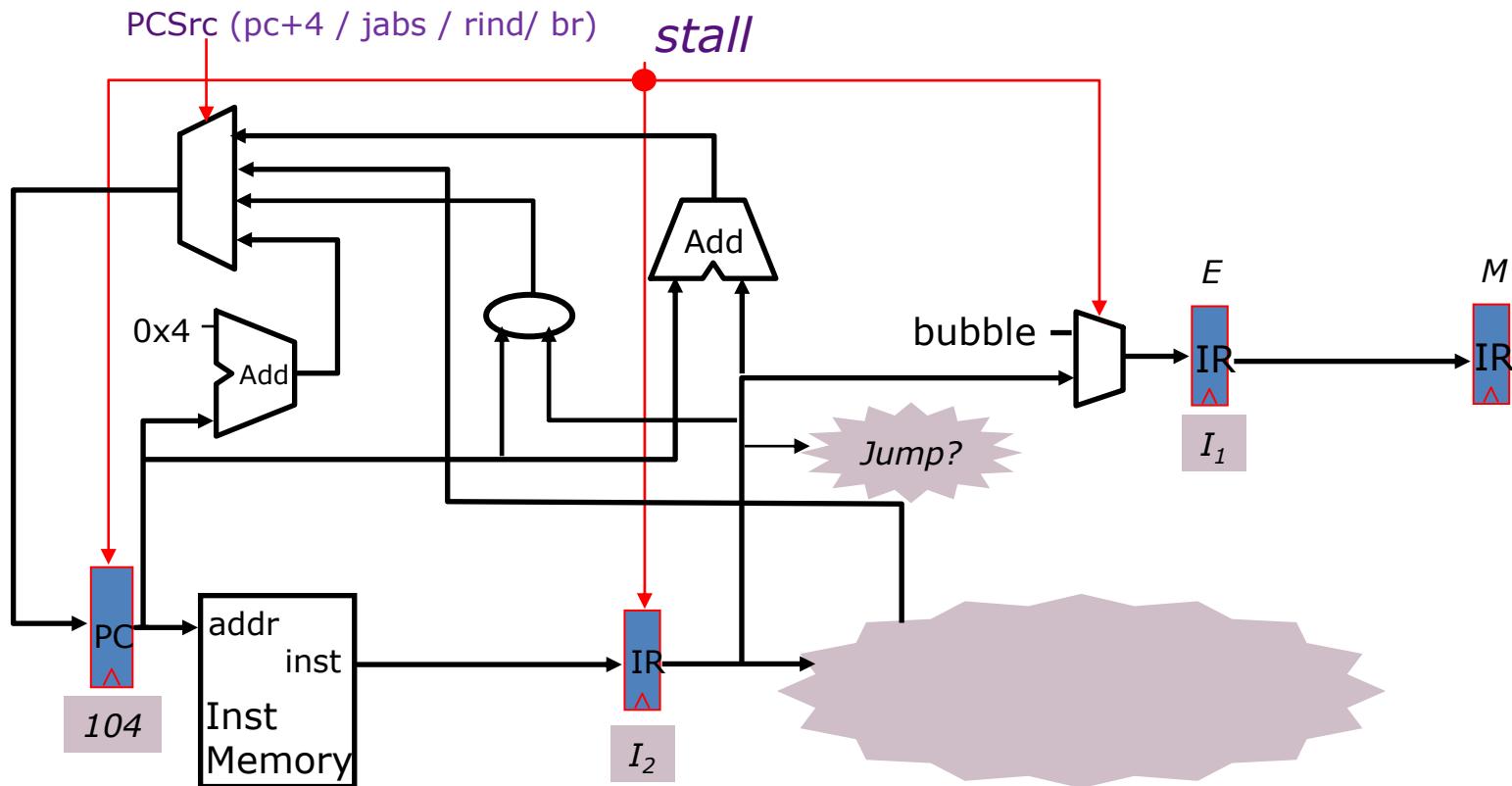
PC Calculation Bubbles

	time									
	t0	t1	t2	t3	t4	t5	t6	t7	...	
(I ₁) $x_1 \leftarrow x_0 + 10$	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁					
(I ₂) $x_3 \leftarrow x_2 + 17$		IF ₂	IF ₂	ID ₂	EX ₂	MA ₂	WB ₂			
(I ₃)			IF ₃	IF ₃	ID ₃	EX ₃	MA ₃	WB ₃		
(I ₄)				IF ₄	IF ₄	ID ₄	EX ₄	MA ₄	WB ₄	

	time									
	t0	t1	t2	t3	t4	t5	t6	t7	...	
IF	I ₁	-	I ₂	-	I ₃	-	I ₄			
ID		I ₁	-	I ₂	-	I ₃	-	I ₄		
EX			I ₁	-	I ₂	-	I ₃	-	I ₄	
MA				I ₁	-	I ₂	-	I ₃	-	I ₄
WB					I ₁	-	I ₂	-	I ₃	-

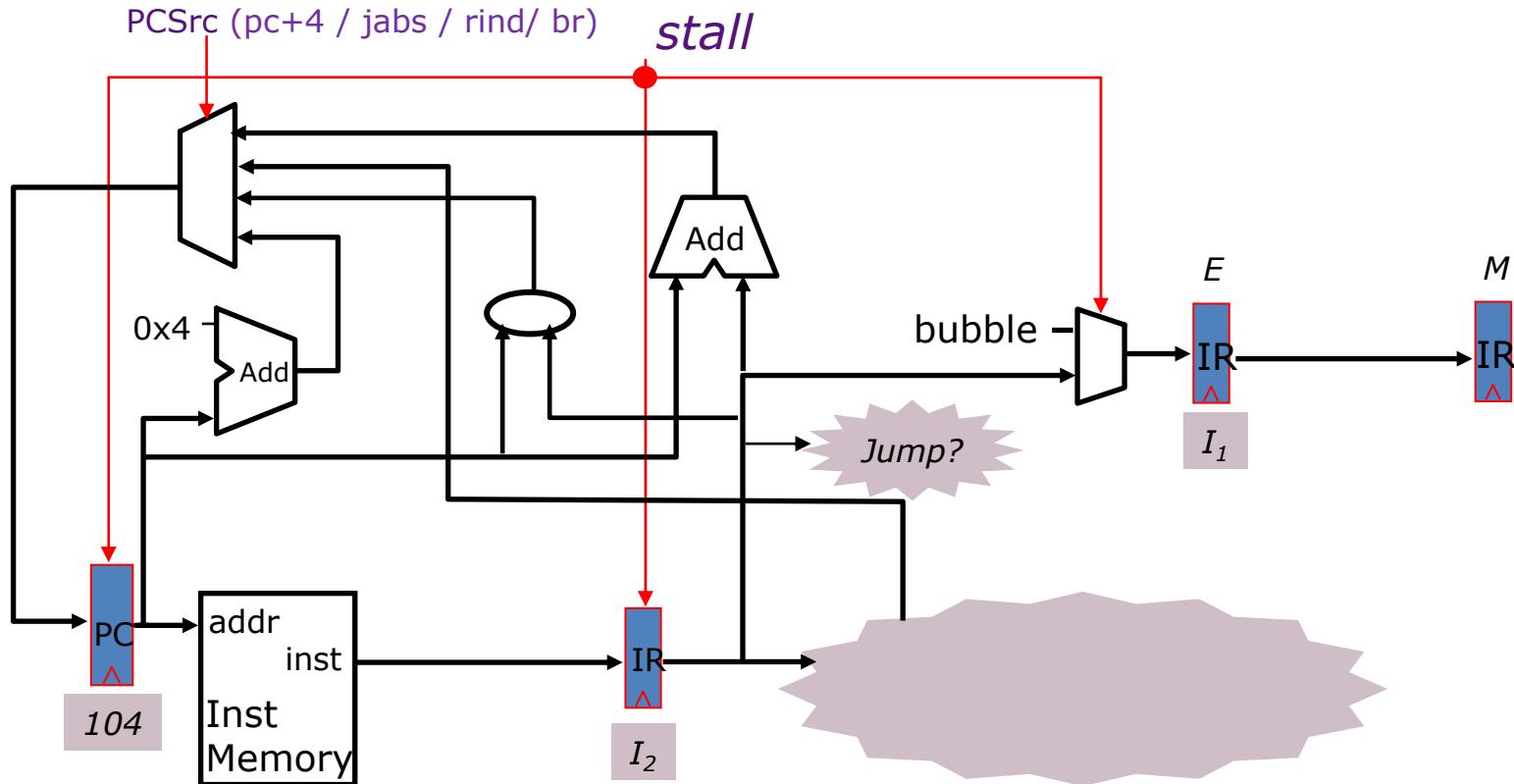
- \Rightarrow pipeline bubble

Спекуляция о том, что следующий адрес это PC+4



I ₁	096	ADD
I ₂	100	J 304
I ₃	104	ADD
I ₄	304	ADD

Спекуляция о том, что следующий адрес это PC+4

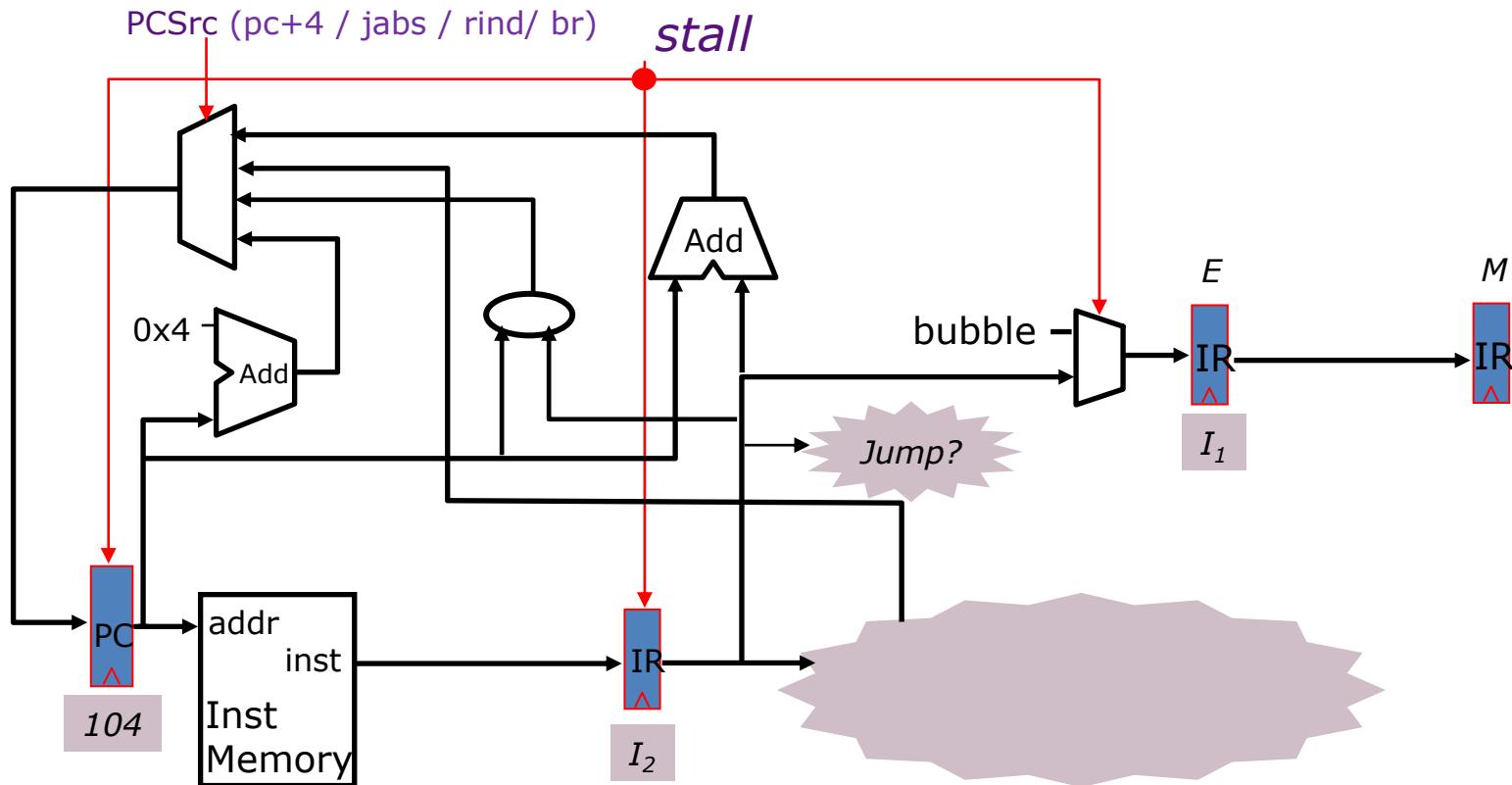


I_1	096	ADD
I_2	100	J 304
I_3	104	ADD
I_4	304	ADD

kill

Команда Jump «убивает» (не блокирует) следующую команду

Спекуляция о том, что следующий адрес это PC+4



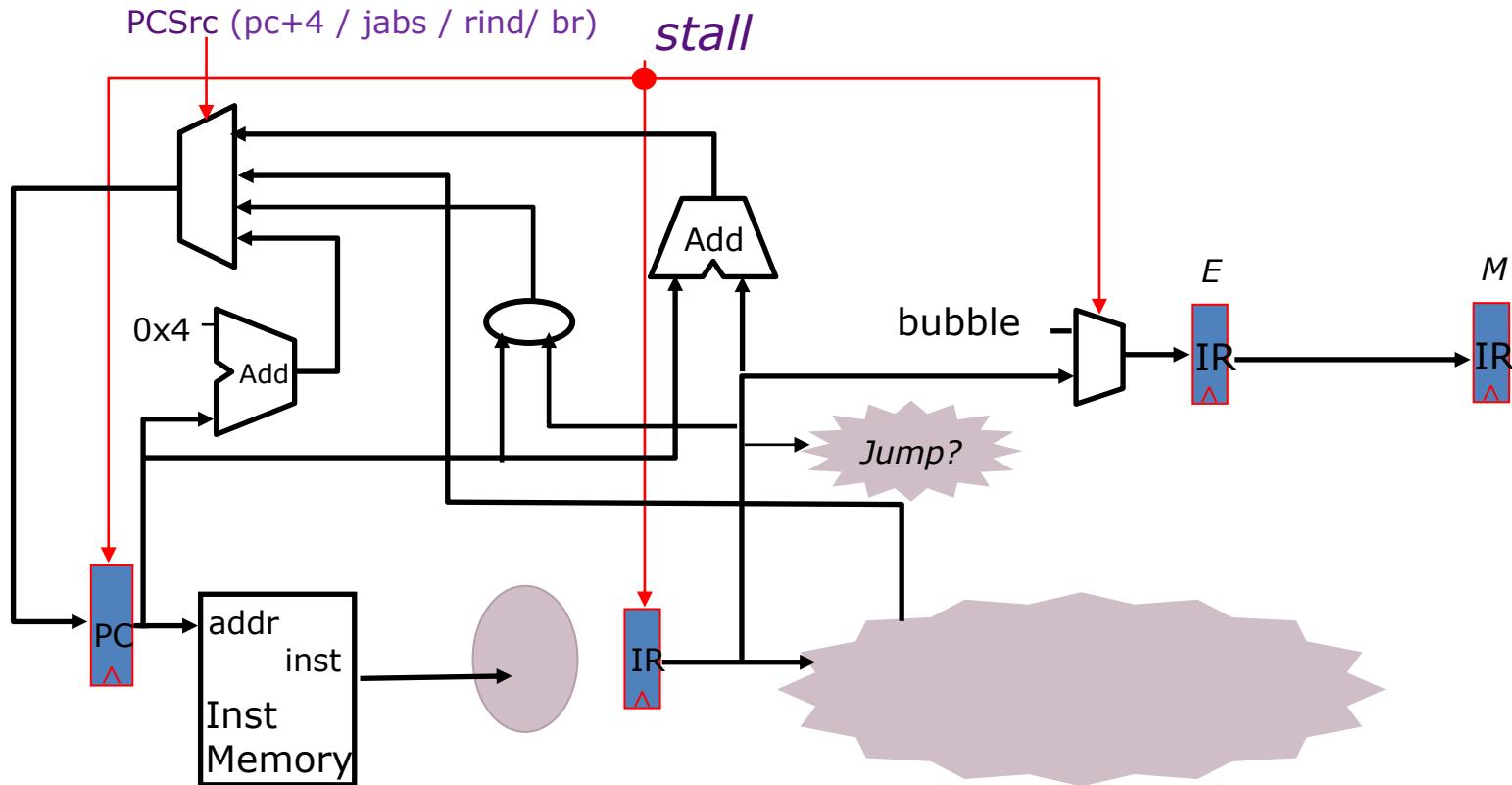
I ₁	096	ADD
I ₂	100	J 304
I ₃	104	ADD
I ₄	304	ADD

kill

Команда Jump «убивает» (не блокирует)
следующую команду

Как?

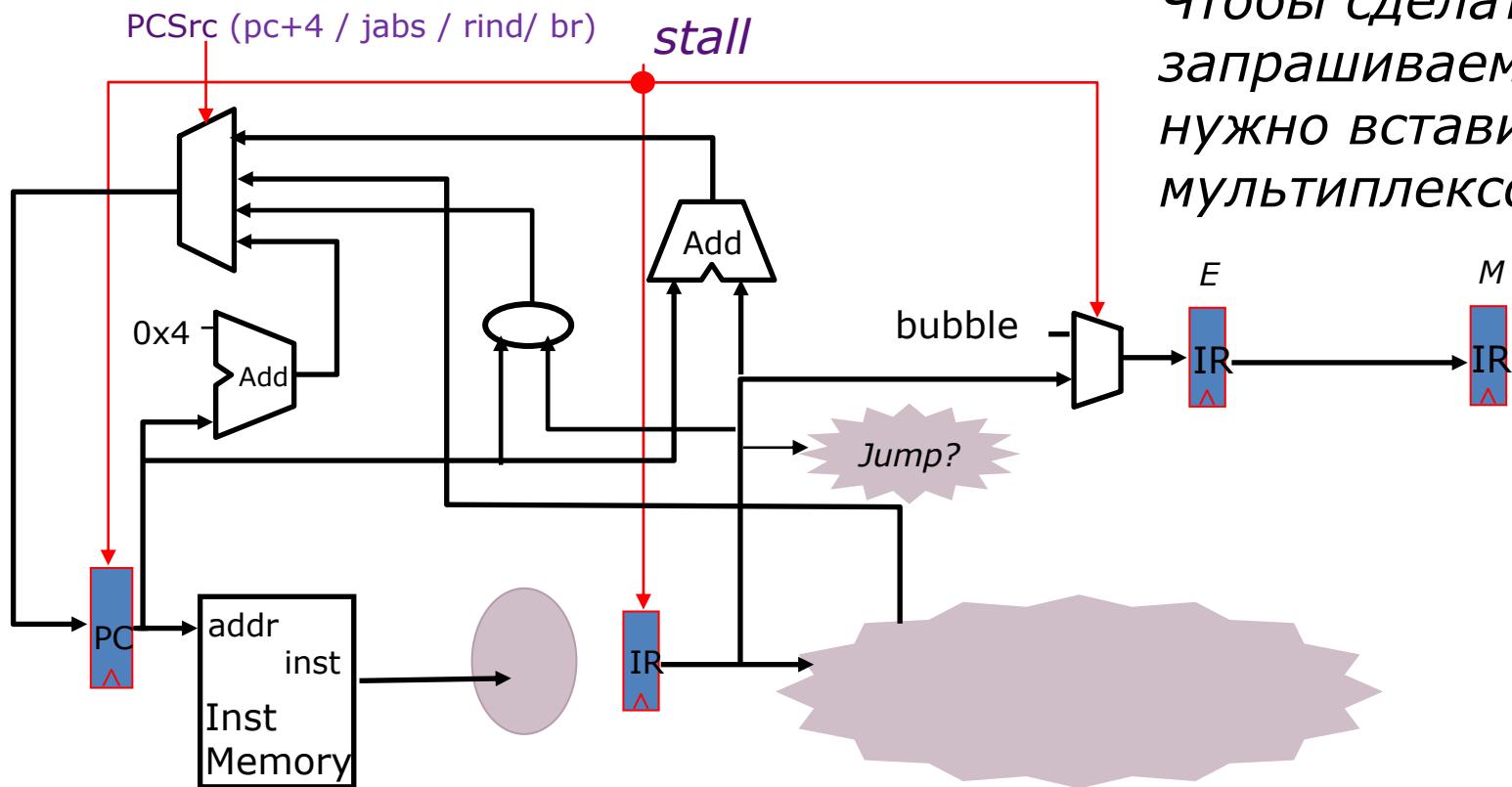
Конвейеризация безусловных переходов



I_1	096	ADD
I_2	100	J 304
I_3	104	ADD
I_4	304	ADD

kill

Конвейеризация безусловных переходов

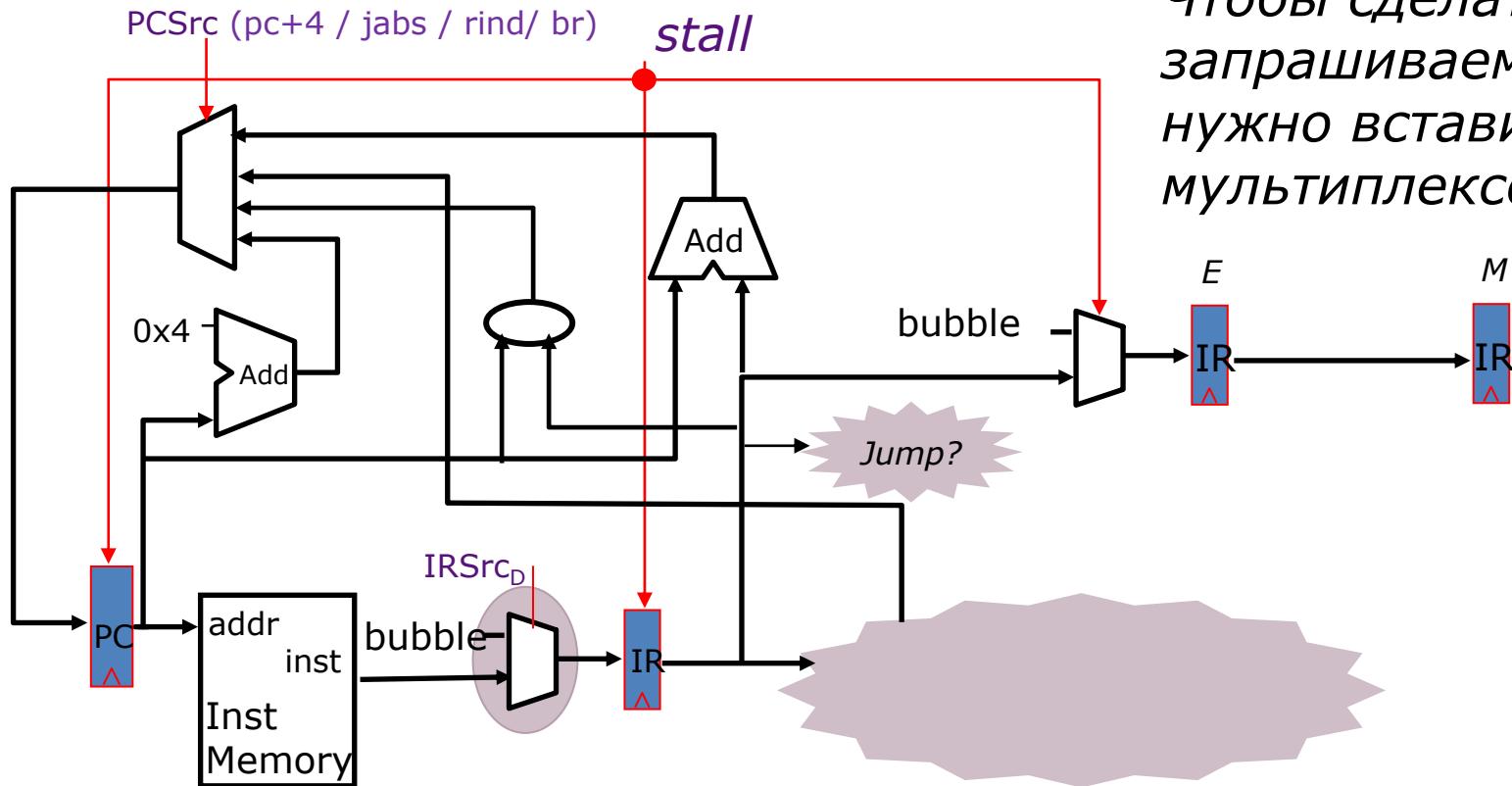


Чтобы сделать kill
запрашиваемой инструкции,
нужно вставить
мультиплексор перед IR

I ₁	096	ADD
I ₂	100	J 304
I ₃	104	ADD
I ₄	304	ADD

kill

Конвейеризация безусловных переходов



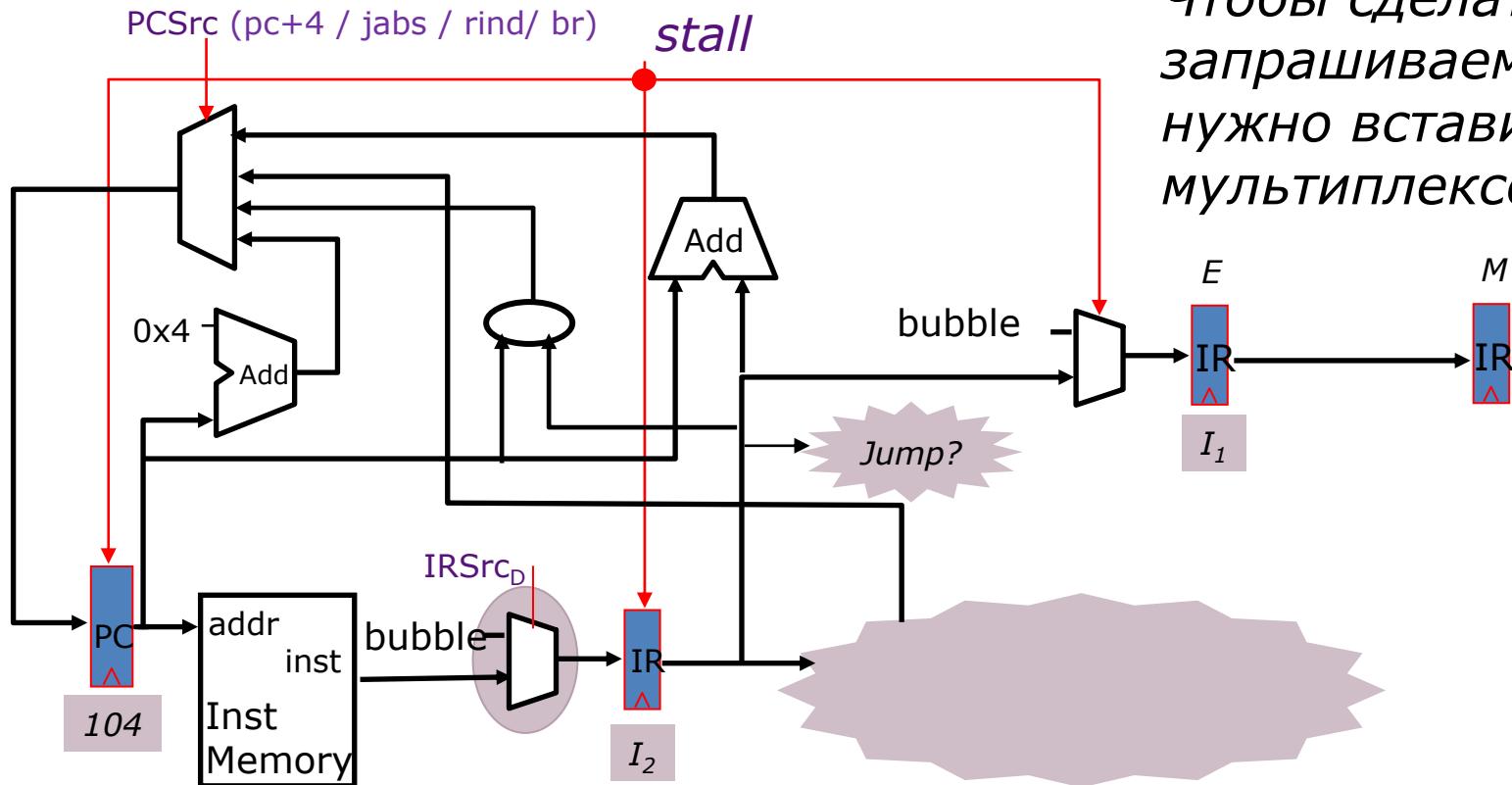
Чтобы сделать kill
запрашиваемой инструкции,
нужно вставить
мультиплексор перед IR

I ₁	096	ADD
I ₂	100	J 304
I ₃	104	ADD
I ₄	304	ADD

kill

IRSrc_D = Case opcode_D
JAL \Rightarrow bubble
... \Rightarrow IM

Конвейеризация безусловных переходов



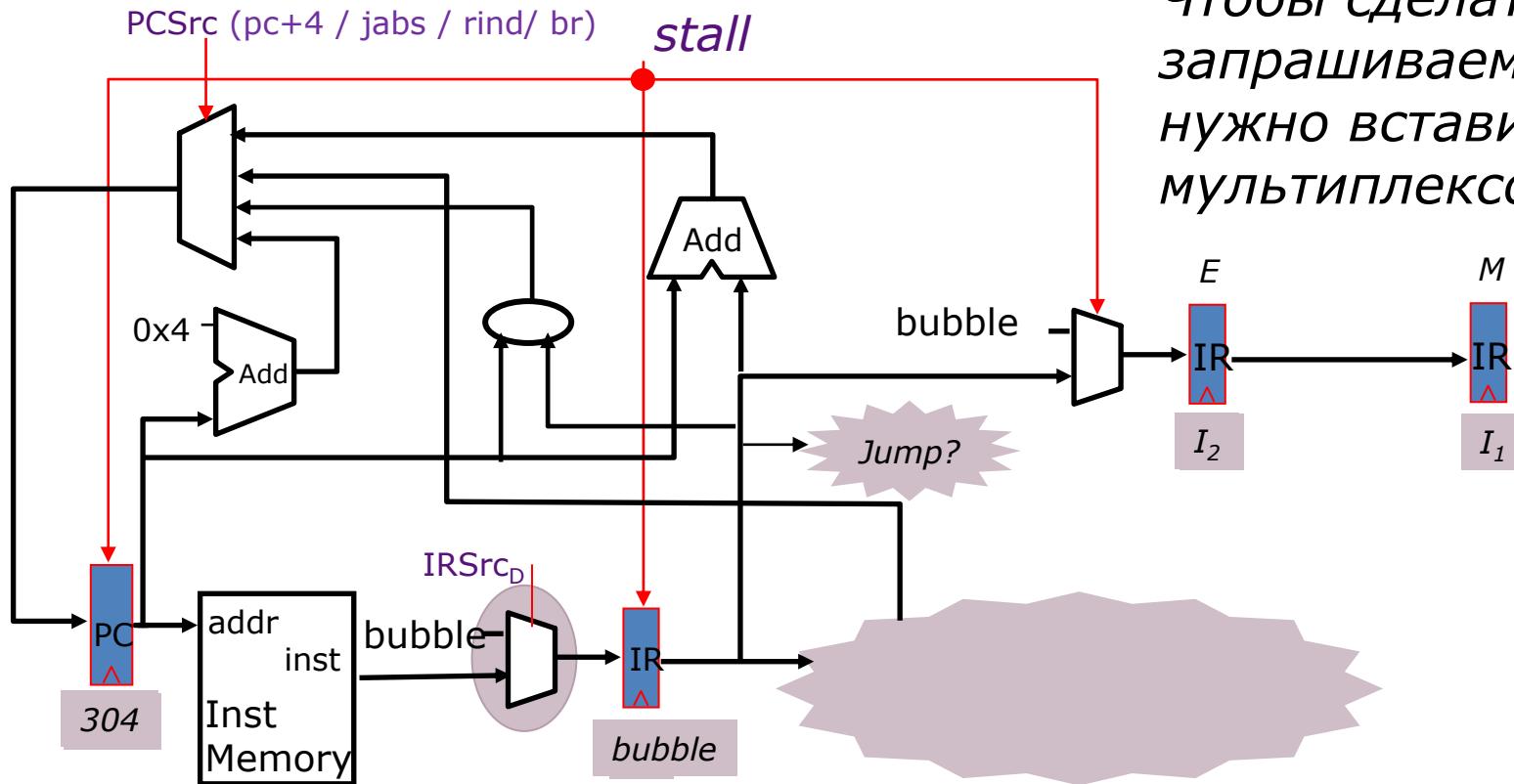
Чтобы сделать kill
запрашиваемой инструкции,
нужно вставить
мультиплексор перед IR

I_1	096	ADD
I_2	100	J 304
I_3	104	ADD
I_4	304	ADD

kill

$IRSrc_D = \begin{cases} \text{Case opcode}_D & \Rightarrow \text{bubble} \\ \text{JAL} & \Rightarrow \text{IM} \\ \dots & \end{cases}$

Конвейеризация безусловных переходов



Чтобы сделать *kill* запрашиваемой инструкции, нужно вставить мультиплексор перед *IR*

I ₁	096	ADD
I ₂	100	J 304
I ₃	104	ADD
I ₄	304	ADD

kill

$IRSrc_D = \begin{cases} \text{Case } opcode_D & \Rightarrow \text{bubble} \\ \text{JAL} & \Rightarrow \text{IM} \\ \dots & \end{cases}$

Диаграмма конвейера при безусловном переходе

time
t0 t1 t2 t3 t4 t5 t6 t7 . . .

Диаграмма конвейера при безусловном переходе

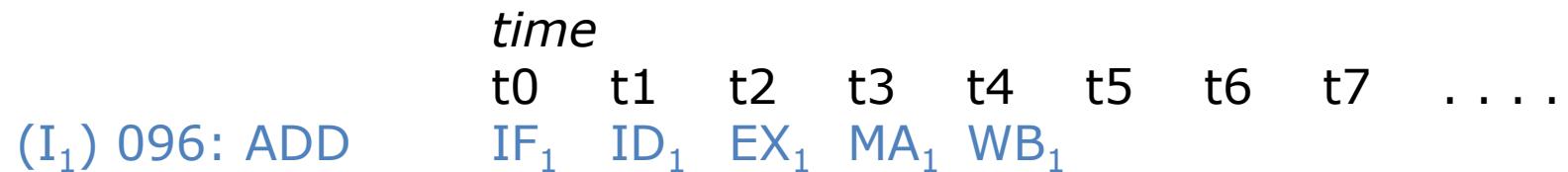


Диаграмма конвейера при безусловном переходе

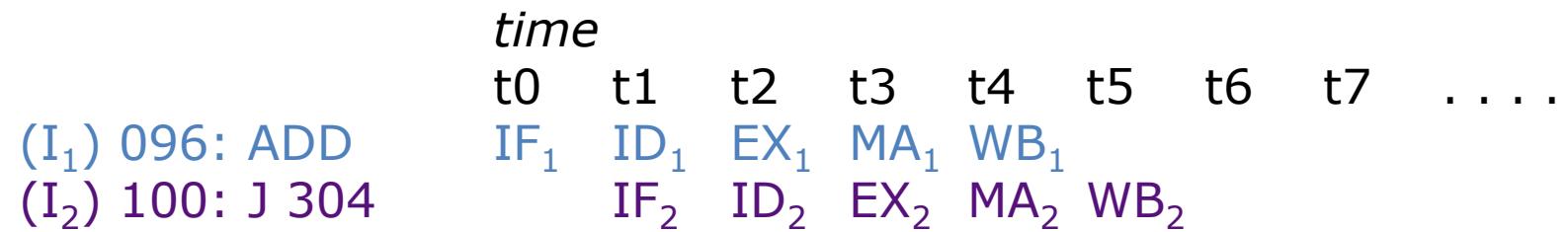


Диаграмма конвейера при безусловном переходе

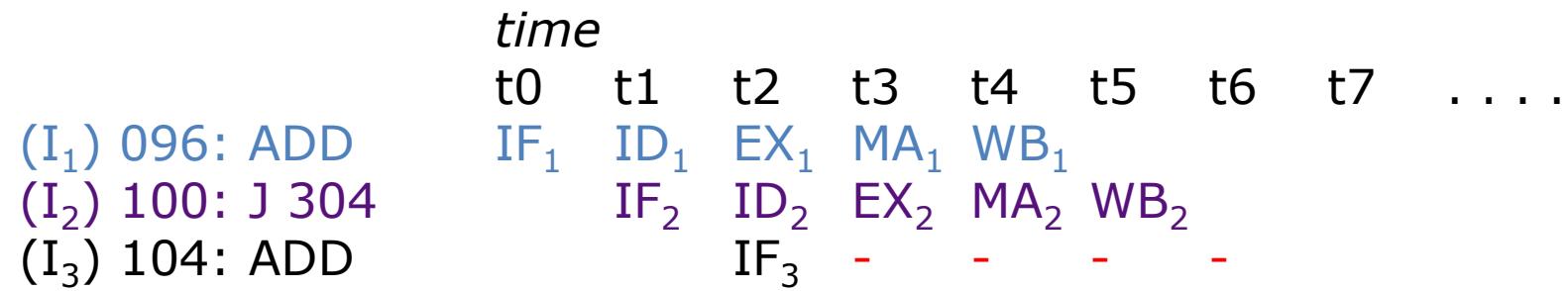


Диаграмма конвейера при безусловном переходе

	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7
(I ₁) 096: ADD	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) 100: J 304		IF ₂	ID ₂	EX ₂	MA ₂	WB ₂			
(I ₃) 104: ADD			IF ₃	-	-	-	-	-	
(I ₄) 304: ADD				IF ₄	ID ₄	EX ₄	MA ₄	WB ₄	

Диаграмма конвейера при безусловном переходе

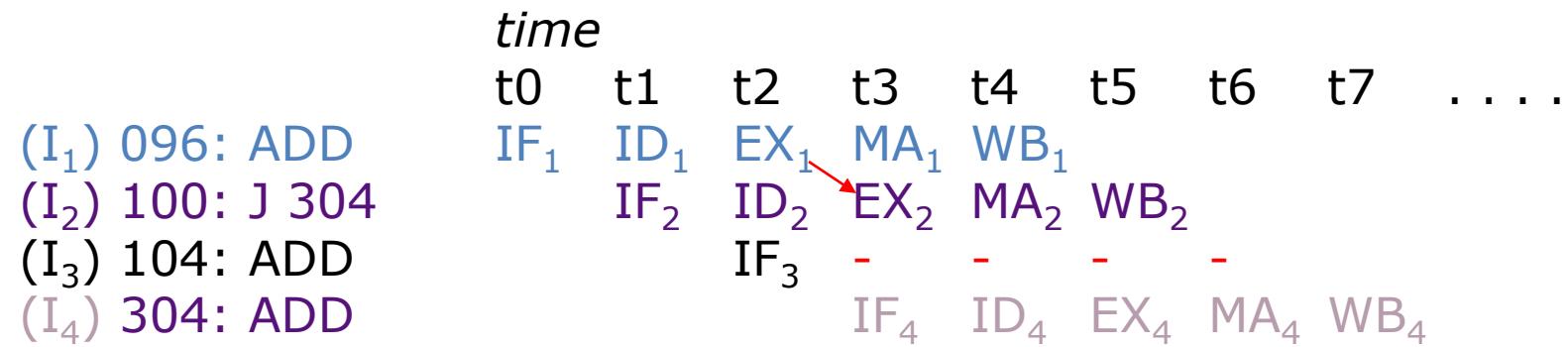
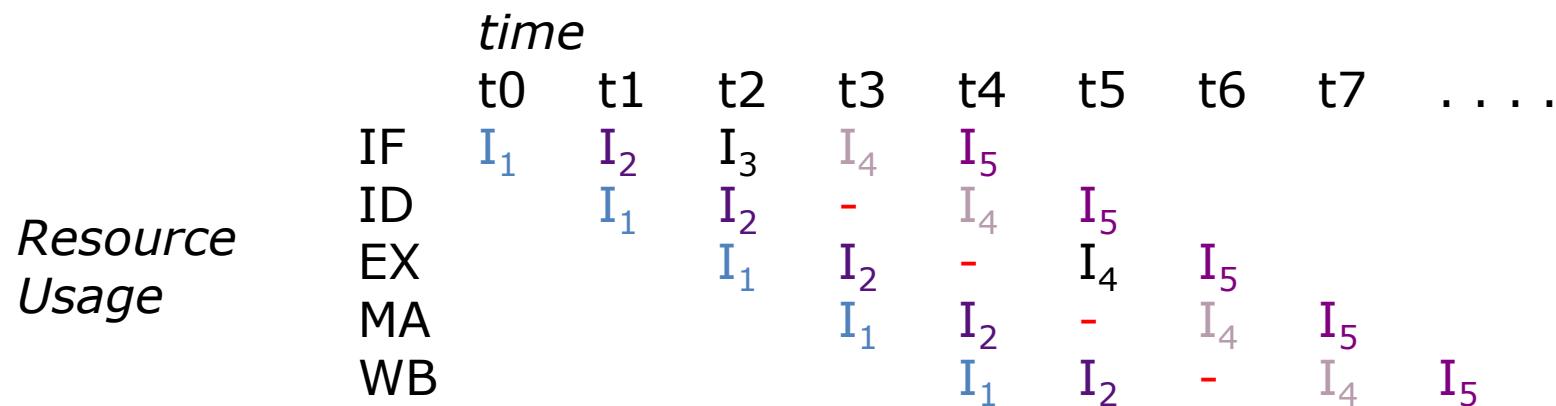
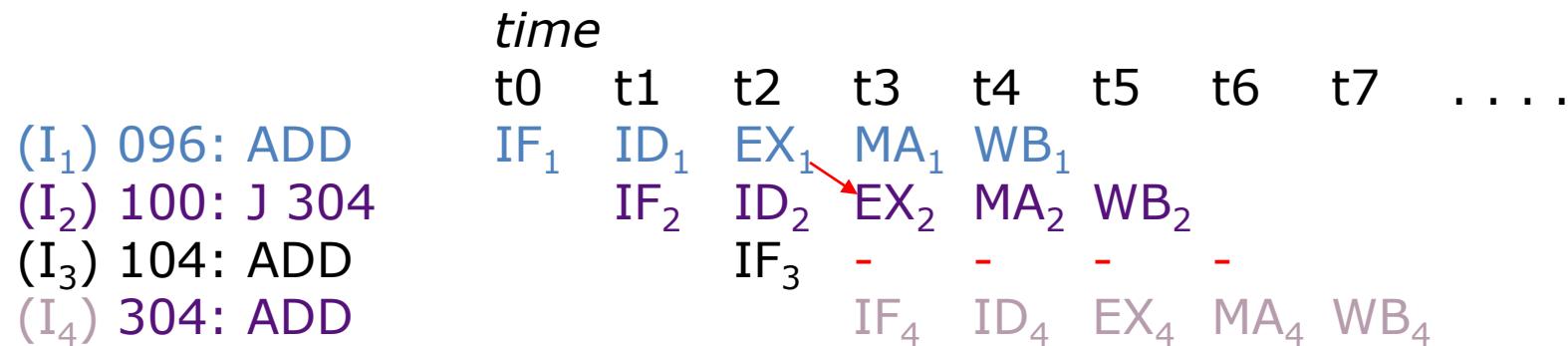
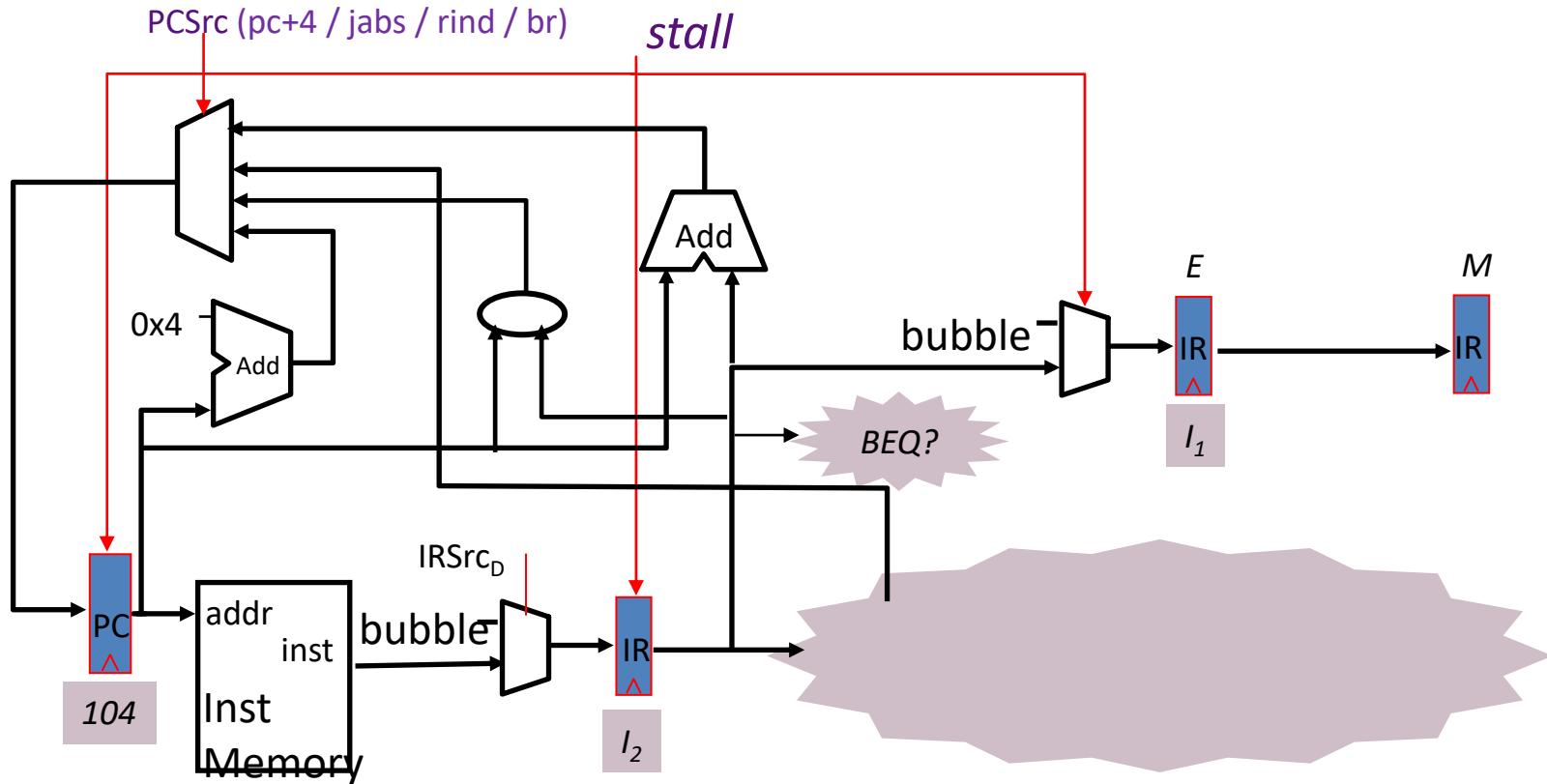


Диаграмма конвейера при безусловном переходе



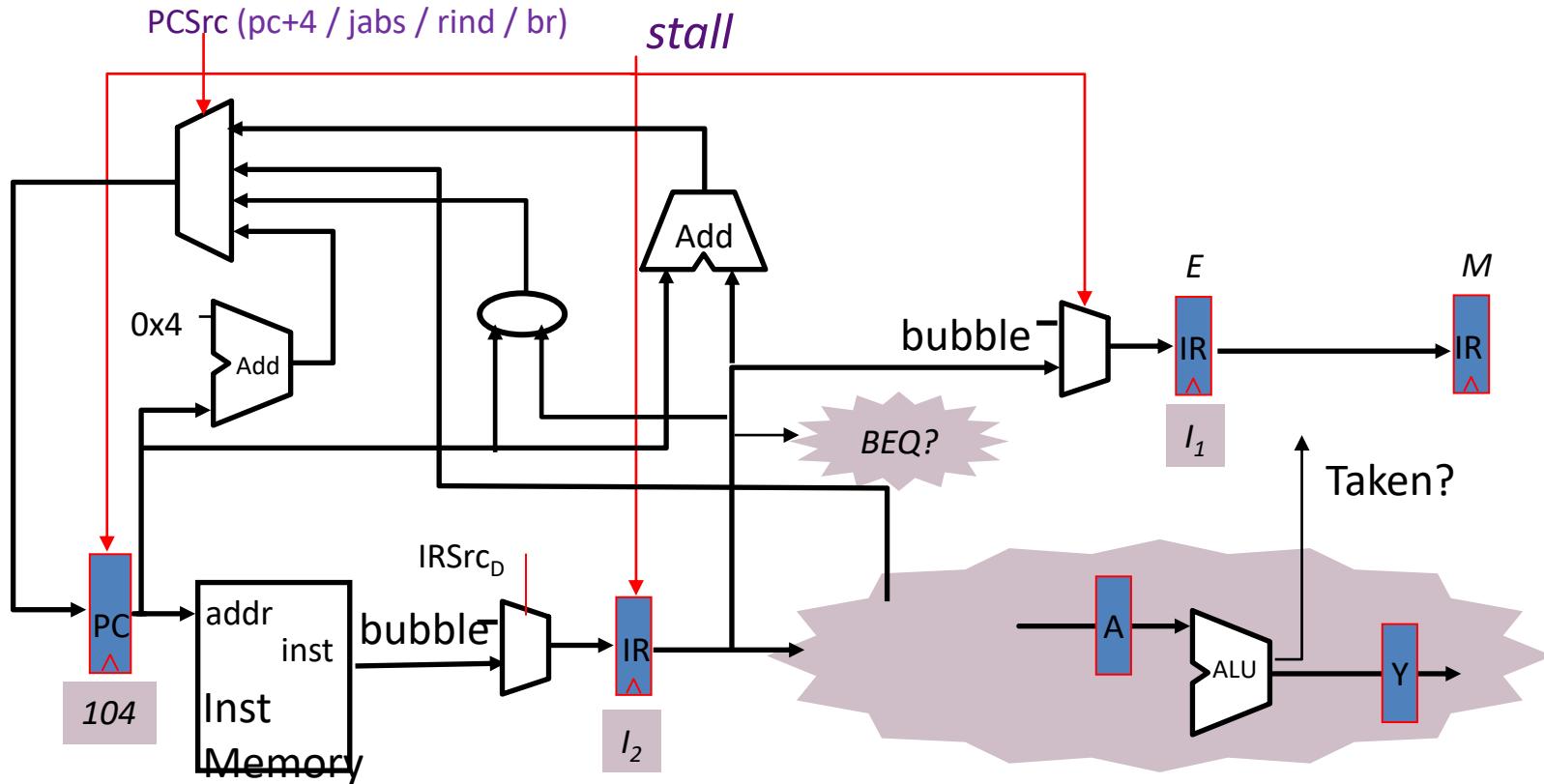
- \Rightarrow *pipeline bubble*

Конвейеризация условных переходов



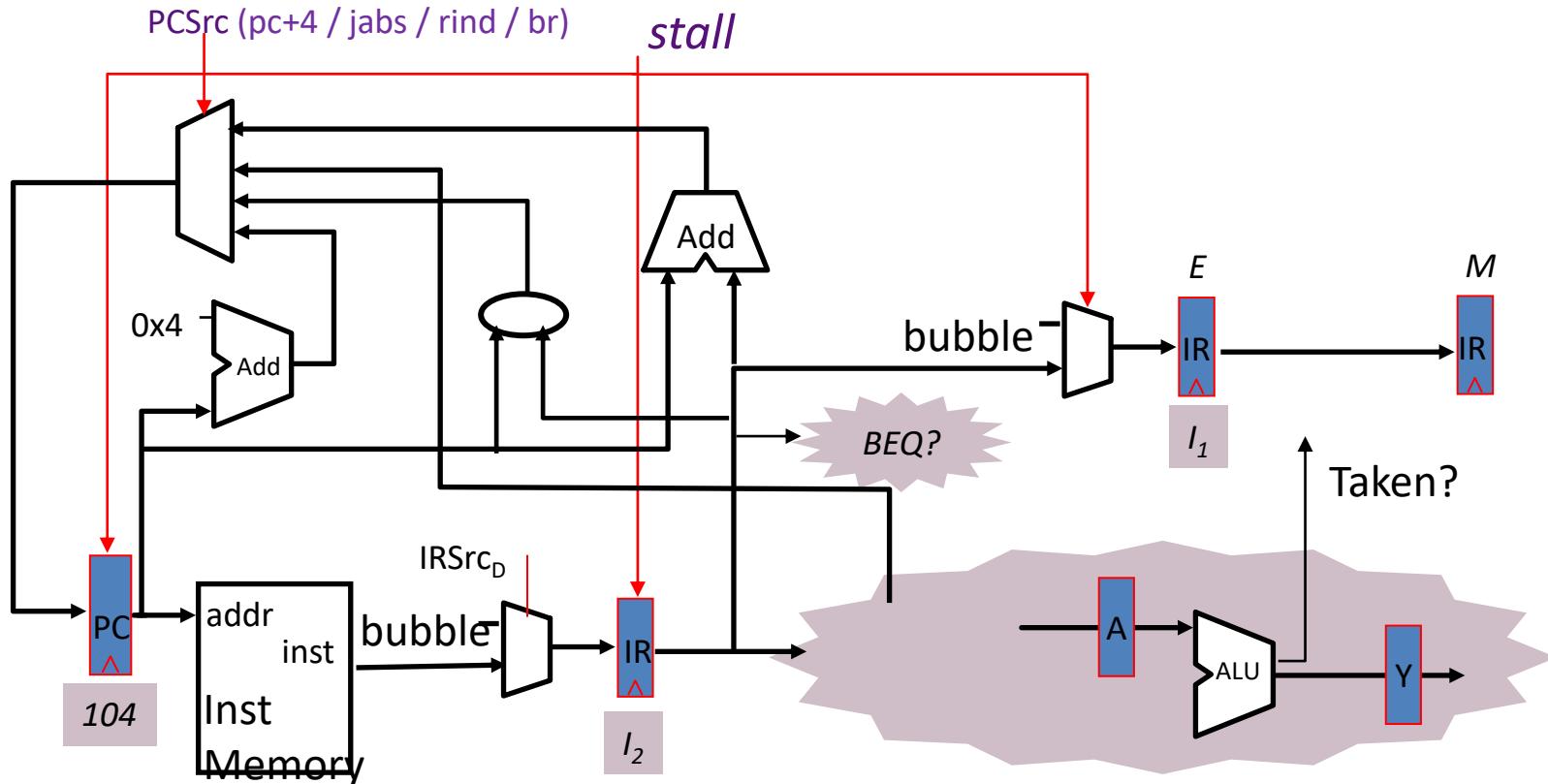
I ₁	096	ADD
I ₂	100	BEQ x1,x2 +200
I ₃	104	ADD
I ₄	304	ADD

Конвейеризация условных переходов



I_1	096	ADD
I_2	100	BEQ $x_1, x_2 + 200$
I_3	104	ADD
I_4	304	ADD

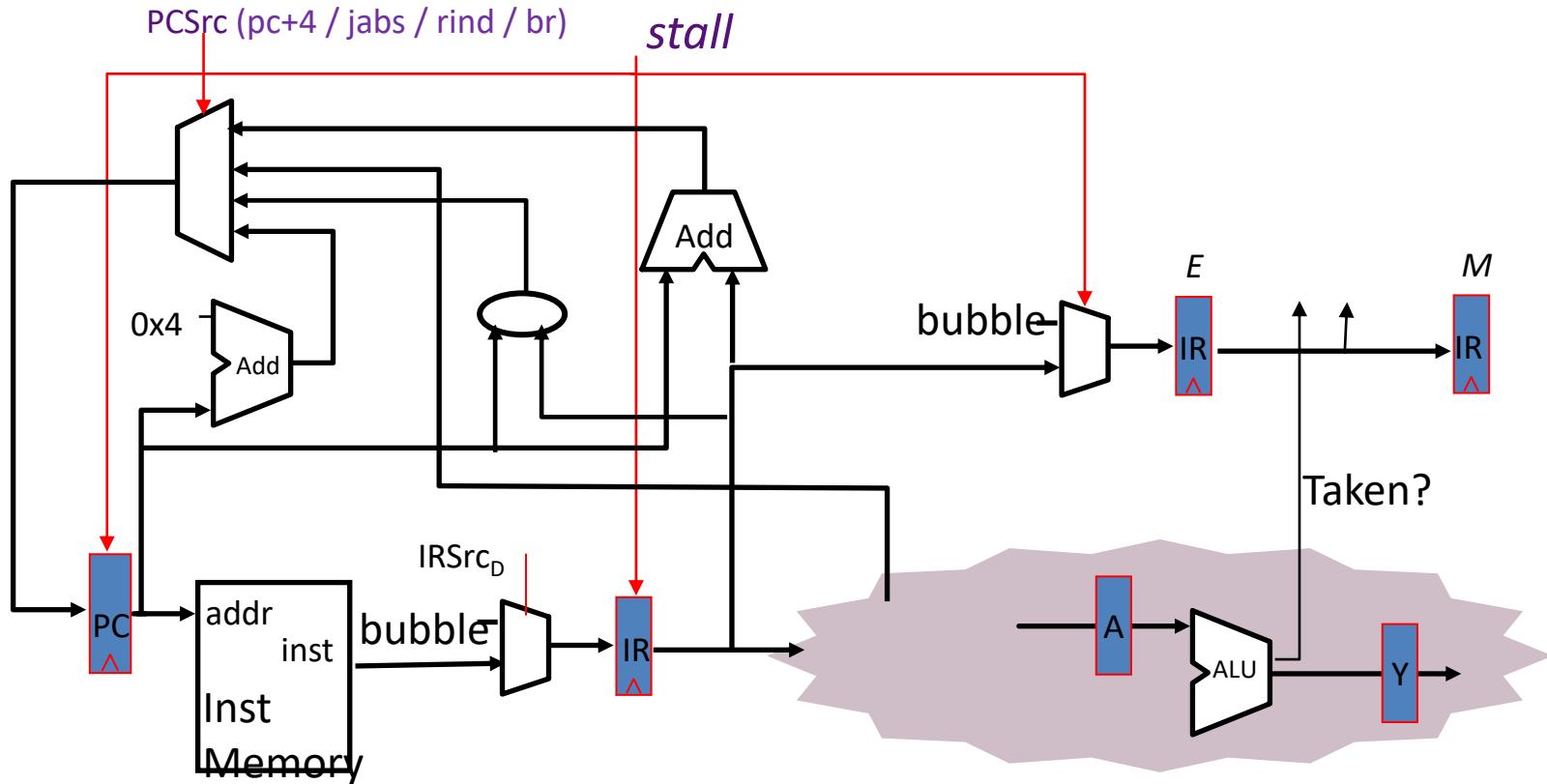
Конвейеризация условных переходов



I_1	096	ADD
I_2	100	BEQ $x_1, x_2 + 200$
I_3	104	ADD
I_4	304	ADD

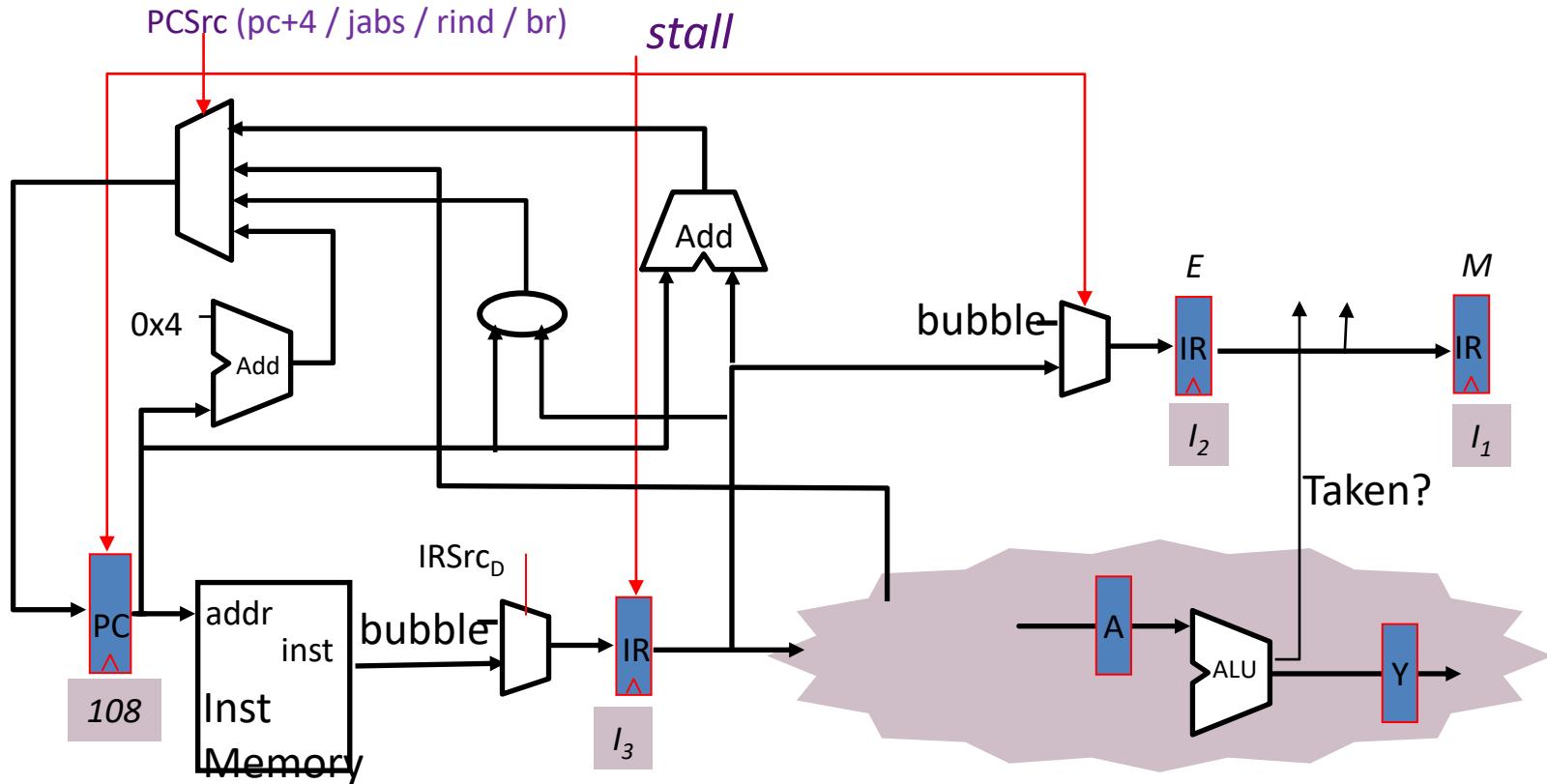
Условие перехода неизвестно вплоть до
стадии Execute
Что нужно предпринять на стадии Decode?

Конвейеризация условных переходов



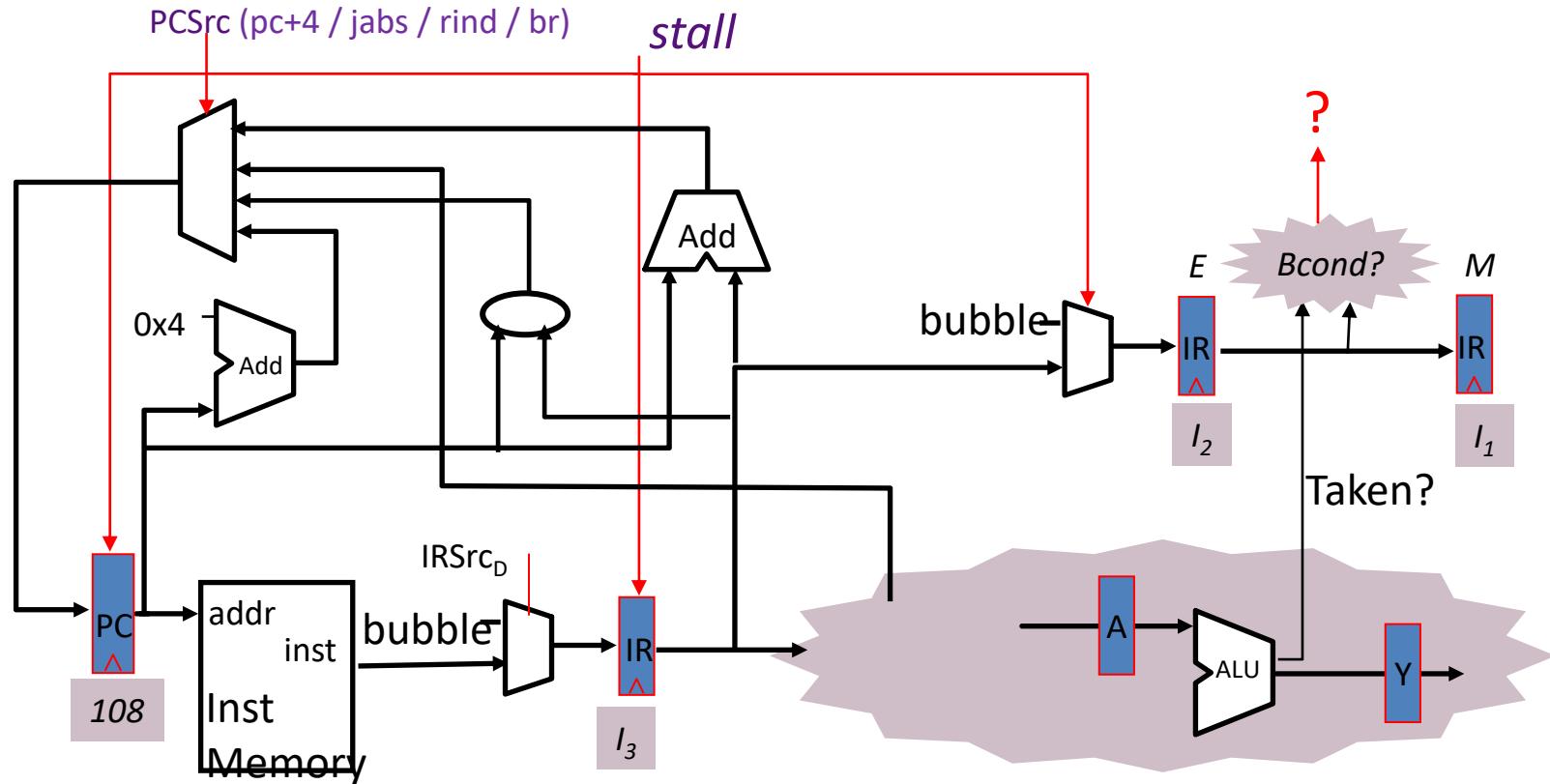
I ₁	096	ADD
I ₂	100	BEQ x1,x2 +200
I ₃	104	ADD
I ₄	304	ADD

Конвейеризация условных переходов



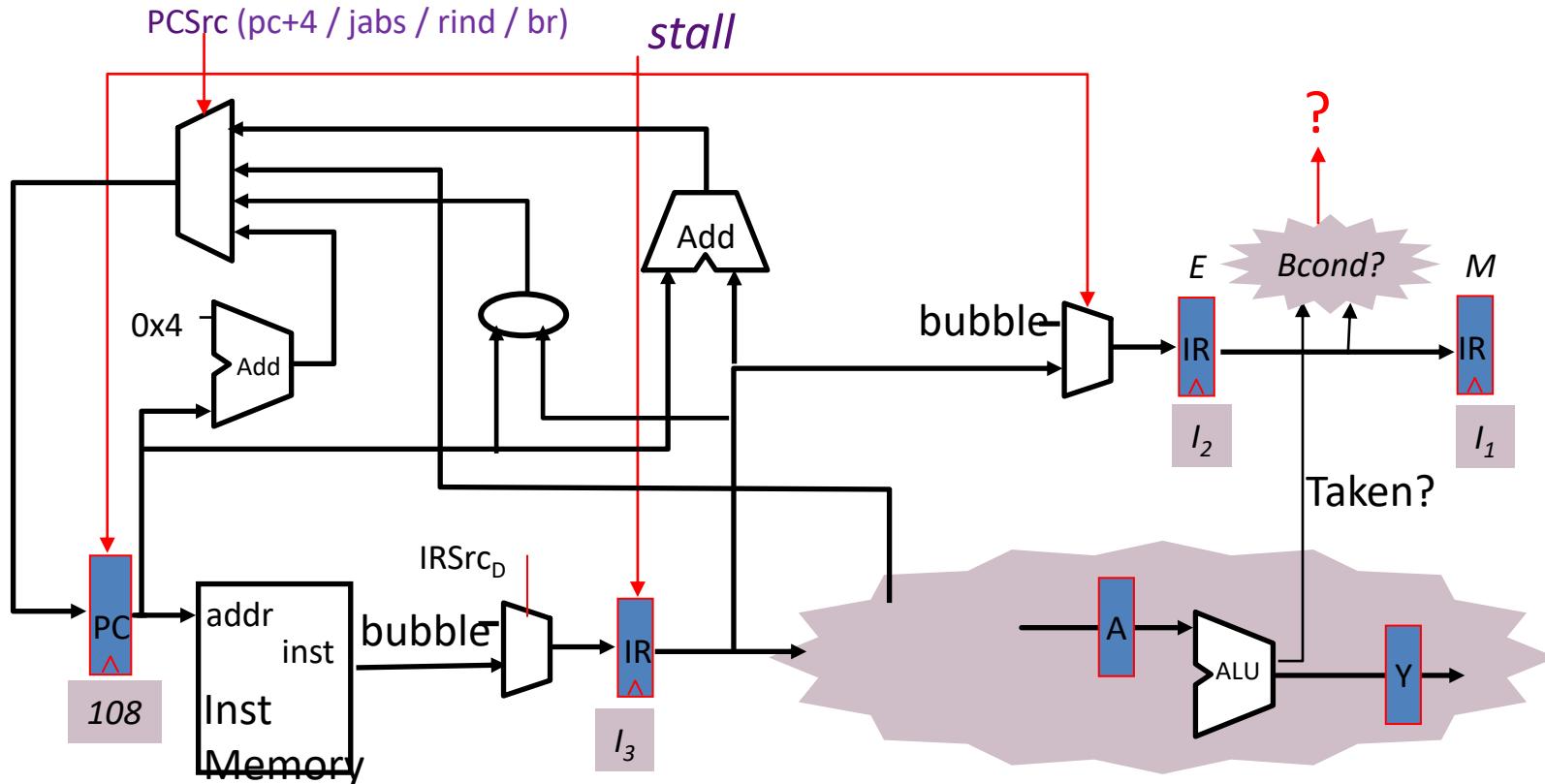
I₁	096	ADD
I₂	100	BEQ x1,x2 +200
I₃	104	ADD
I₄	304	ADD

Конвейеризация условных переходов



I ₁	096	ADD
I ₂	100	BEQ x1,x2 +200
I ₃	104	ADD
I ₄	304	ADD

Конвейеризация условных переходов

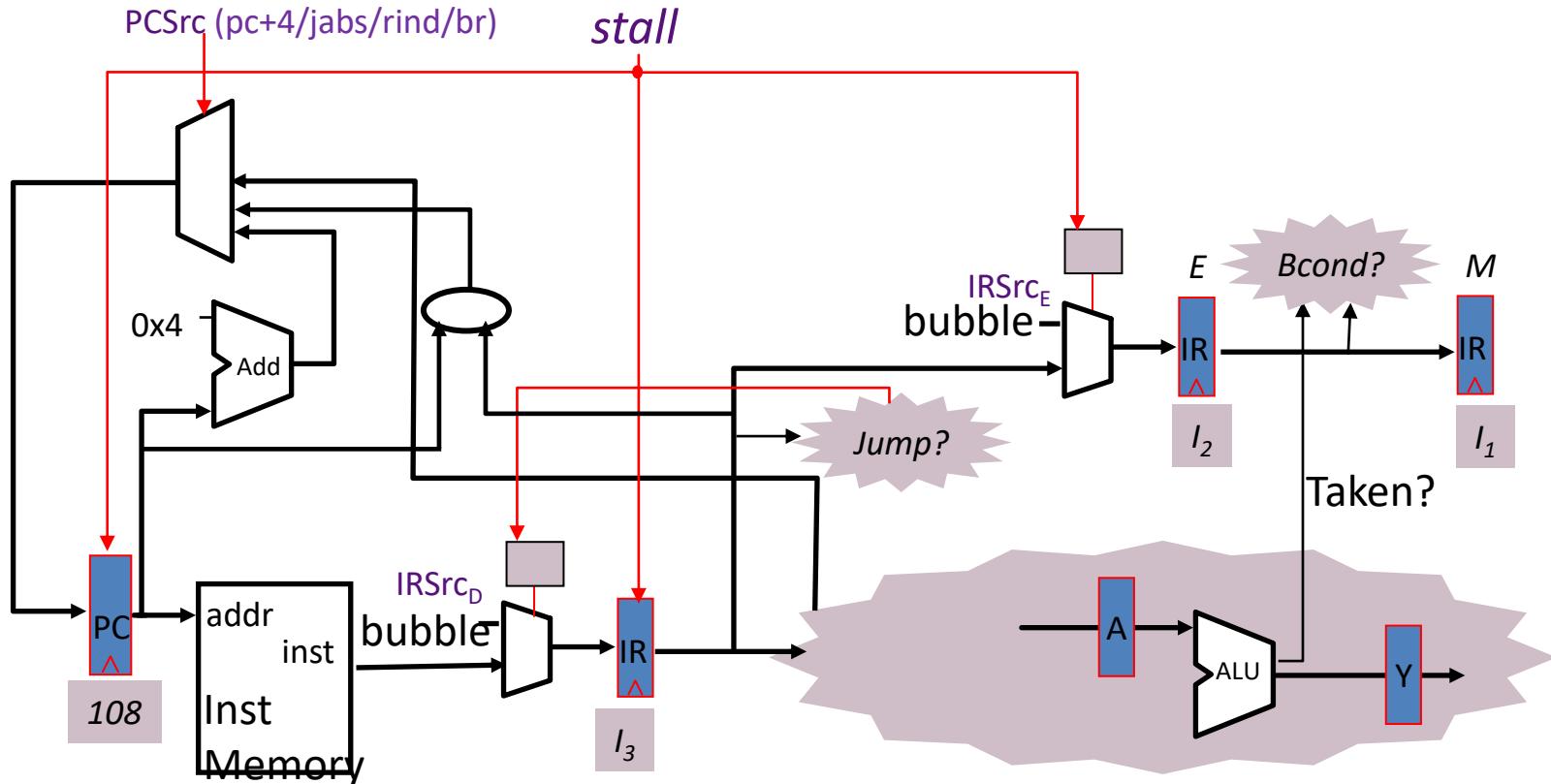


I ₁	096	ADD
I ₂	100	BEQ x1,x2 +200
I ₃	104	ADD
I ₄	304	ADD

Если совершен переход:

- «Убить» две следующие инструкции
- Инструкция на стадии Decode не валидна => **сигнал блокировки не валиден**

Конвейеризация условных переходов

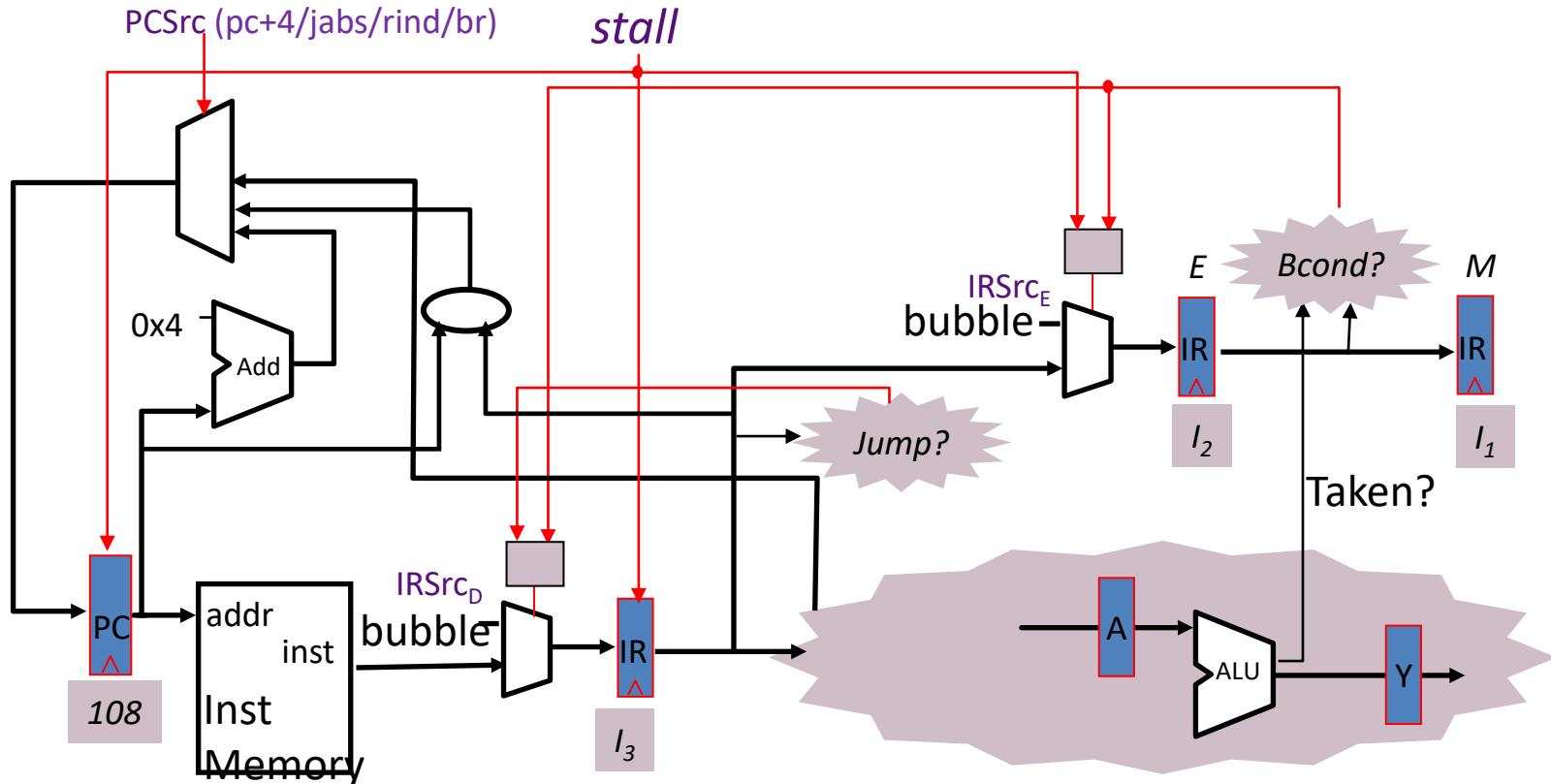


I ₁ :	096	ADD
I ₂ :	100	BEQ x1,x2 +200
I ₃ :	104	ADD
I ₄ :	304	ADD

Если совершен переход:

- «Убить» две следующие инструкции
- Инструкция на стадии Decode не валидна => **сигнал блокировки не валиден**

Конвейеризация условных переходов

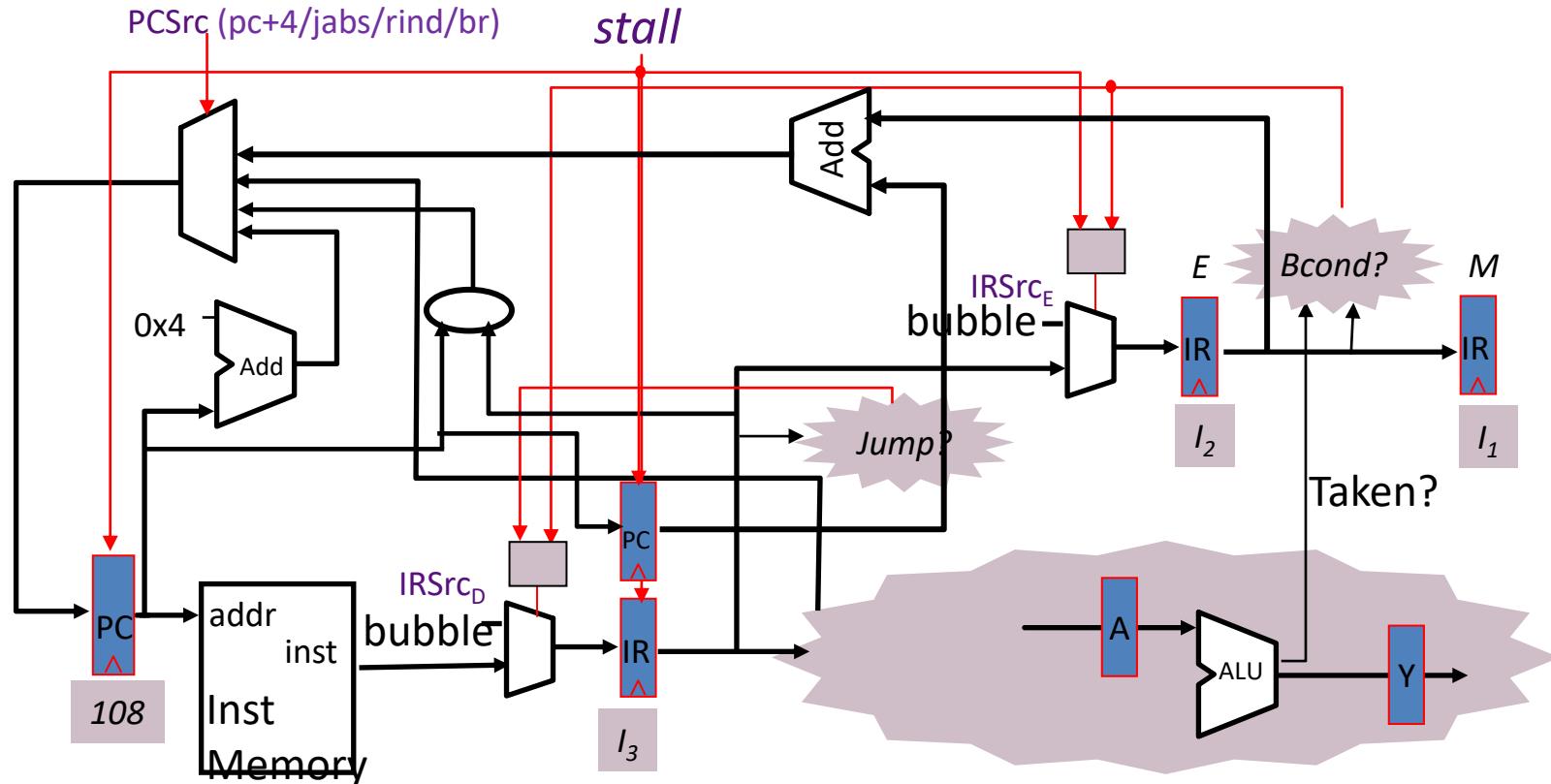


I ₁ :	096	ADD
I ₂ :	100	BEQ x1,x2 +200
I ₃ :	104	ADD
I ₄ :	304	ADD

Если совершен переход:

- «Убить» две следующие инструкции
- Инструкция на стадии Decode не валидна => **сигнал блокировки не валиден**

Конвейеризация условных переходов



I ₁ :	096	ADD
I ₂ :	100	BEQ x1,x2 +200
I ₃ :	104	ADD
I ₄ :	304	ADD

Если совершен переход:

- «Убить» две следующие инструкции
- Инструкция на стадии Decode не валидна => **сигнал блокировки не валиден**

Диаграмма конвейера при условном переходе

(разрешен на стадии Execute)

	<i>time</i>									
	t0	t1	t2	t3	t4	t5	t6	t7	...	
(I ₁) 096: ADD		IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) 100: BEQ	+200		IF ₂	ID ₂	EX ₂	MA ₂	WB ₂			
(I ₃) 104: ADD			IF ₃	ID ₃						
(I ₄) 108:				IF ₄						
(I ₅) 304: ADD					IF ₅	ID ₅	EX ₅	MA ₅	WB ₅	

	<i>time</i>									
	t0	t1	t2	t3	t4	t5	t6	t7	...	
Resource Usage	IF	I ₁	I ₂	I ₃	I ₄	I ₅				
	ID		I ₁	I ₂	I ₃	-	I ₅			
	EX			I ₁	I ₂	-	-	I ₅		
	MA				I ₁	I ₂	-	-	I ₅	
	WB					I ₁	I ₂	-	-	I ₅

- ⇒ *pipeline bubble*

Диаграмма конвейера при условном переходе

(разрешен на стадии Execute)

time
t0 t1 t2 t3 t4 t5 t6 t7 . . .

Диаграмма конвейера при условном переходе

(разрешен на стадии Execute)

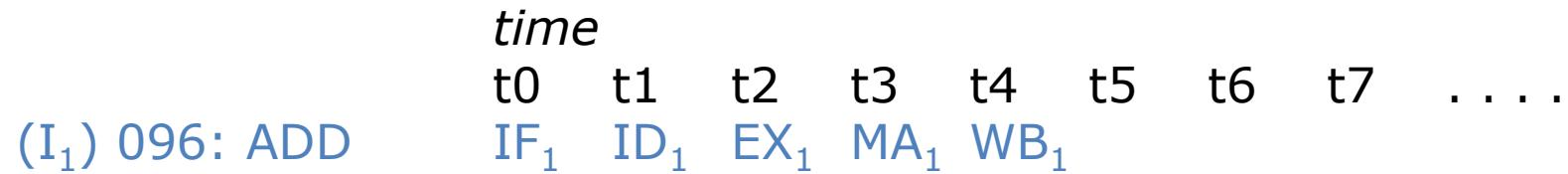


Диаграмма конвейера при условном переходе

(разрешен на стадии Execute)

	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) 096: ADD	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) 100: BEQ +200	IF ₂	ID ₂	EX ₂	MA ₂	WB ₂				

Диаграмма конвейера при условном переходе

(разрешен на стадии Execute)

	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) 096: ADD		IF ₁	ID ₁	EX ₁	MA ₁	WB ₁			
(I ₂) 100: BEQ	+200		IF ₂	ID ₂	EX ₂	MA ₂	WB ₂		
(I ₃) 104: ADD			IF ₃	ID ₃	-	-	-		

Диаграмма конвейера при условном переходе

(разрешен на стадии Execute)

	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) 096: ADD		IF ₁	ID ₁	EX ₁	MA ₁	WB ₁			
(I ₂) 100: BEQ	+200		IF ₂	ID ₂	EX ₂	MA ₂	WB ₂		
(I ₃) 104: ADD			IF ₃	ID ₃	-	-	-		
(I ₄) 108:				IF ₄	-	-	-	-	

Диаграмма конвейера при условном переходе (разрешен на стадии Execute)

	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) 096: ADD		IF ₁	ID ₁	EX ₁	MA ₁	WB ₁			
(I ₂) 100: BEQ	+200		IF ₂	ID ₂	EX ₂	MA ₂	WB ₂		
(I ₃) 104: ADD			IF ₃	ID ₃	-	-	-		
(I ₄) 108:				IF ₄	-	-	-	-	
(I ₅) 304: ADD					IF ₅	ID ₅	EX ₅	MA ₅	WB ₅

Диаграмма конвейера при условном переходе (разрешен на стадии Execute)

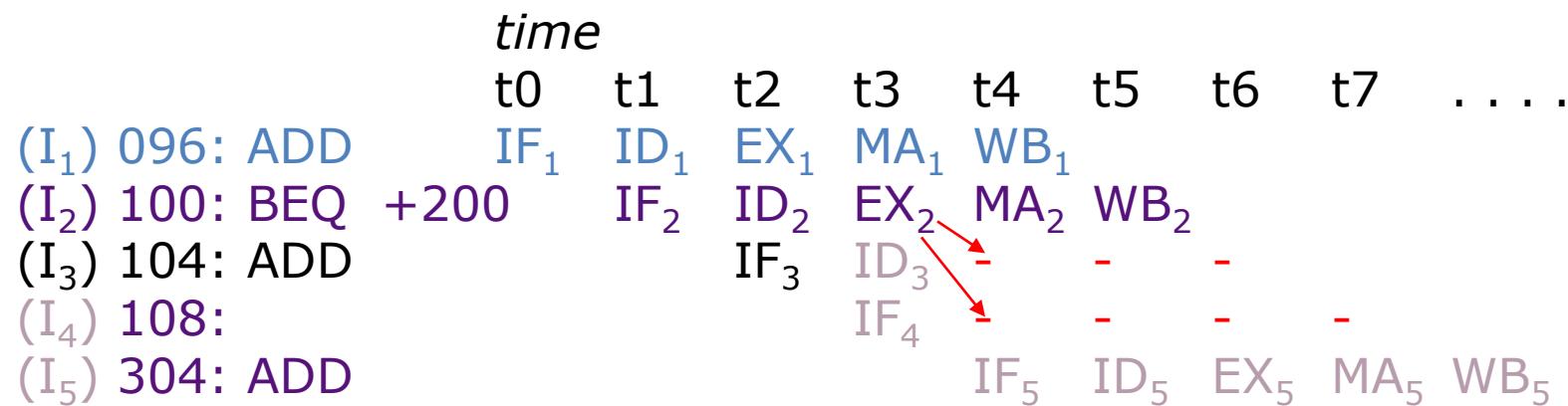


Диаграмма конвейера при условном переходе

(разрешен на стадии Execute)

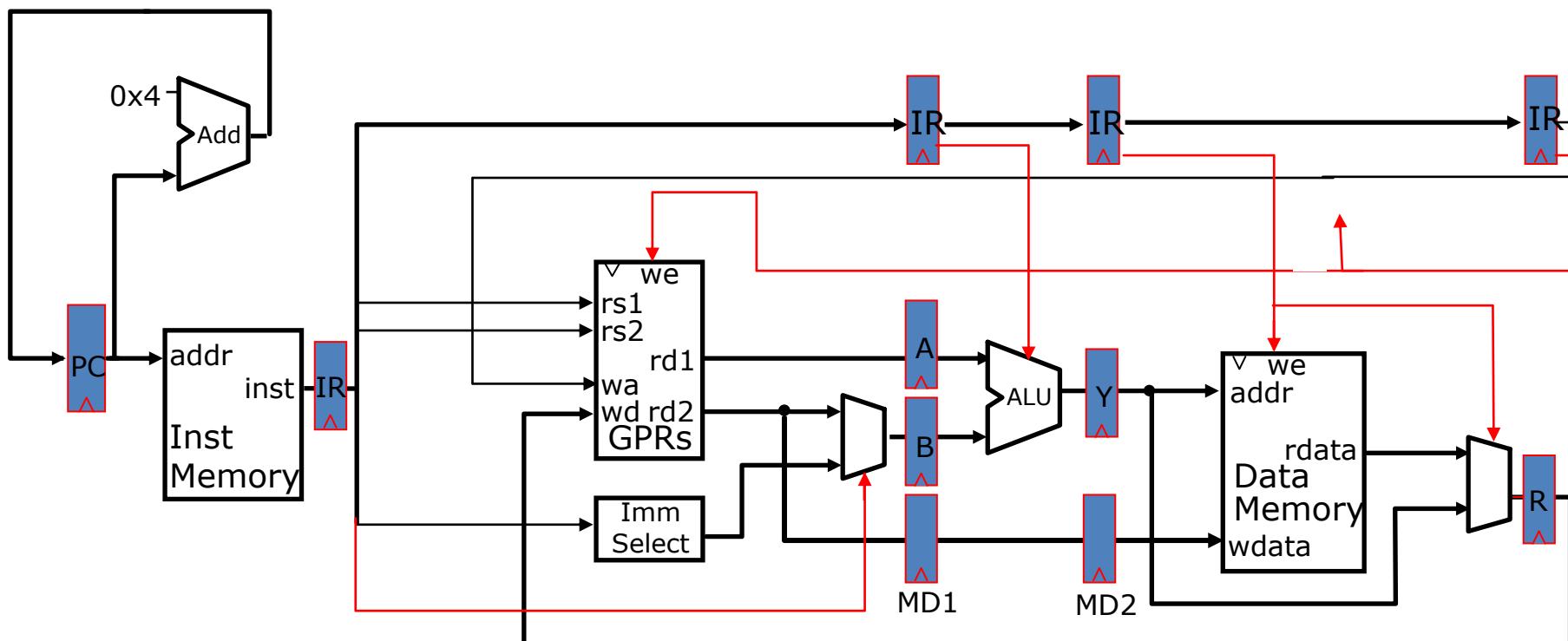
	<i>time</i>									
	t0	t1	t2	t3	t4	t5	t6	t7	...	
(I ₁) 096: ADD		IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) 100: BEQ	+200		IF ₂	ID ₂	EX ₂	MA ₂	WB ₂			
(I ₃) 104: ADD			IF ₃	ID ₃						
(I ₄) 108:				IF ₄						
(I ₅) 304: ADD					IF ₅	ID ₅	EX ₅	MA ₅	WB ₅	

	<i>time</i>									
	t0	t1	t2	t3	t4	t5	t6	t7	...	
Resource Usage	IF	I ₁	I ₂	I ₃	I ₄	I ₅				
	ID		I ₁	I ₂	I ₃	-	I ₅			
	EX			I ₁	I ₂	-	-	I ₅		
	MA				I ₁	I ₂	-	-	I ₅	
	WB					I ₁	I ₂	-	-	I ₅

- ⇒ *pipeline bubble*

Что, если...

- Что, если бы мы использовали простое условие перехода, в котором только один регистр rs1 сравнивается с нулем?
- Можно ли сделать лучше?



Простые условные переходы

сравнение на стадии Decode (сравнение только одного регистра с нулем)

	<i>time</i>									
	t0	t1	t2	t3	t4	t5	t6	t7	...	
(I ₁) 096: ADD		IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) 100: BEQZ +200			IF ₂	ID ₂	EX ₂	MA ₂	WB ₂			
(I ₃) 104: ADD				IF ₃	-	-	-	-		
(I ₄) 300: ADD					IF ₄	ID ₄	EX ₄	MA ₄	WB ₄	

<i>Resource Usage</i>	<i>time</i>									
	t0	t1	t2	t3	t4	t5	t6	t7	...	
	IF	I ₁	I ₂	I ₃	I ₄	I ₅				
	ID		I ₁	I ₂	-	I ₄	I ₅			
	EX			I ₁	I ₂	-	I ₄	I ₅		
	MA				I ₁	I ₂	-	I ₄	I ₅	
	WB					I ₁	I ₂	-	I ₄	I ₅

- ⇒ *pipeline bubble*

Что, если... (2)

- Что, если бы вместо «очистки» конвейера после коллизии по управлению мы могли бы найти способ сделать инструкцию, следующую за инструкцией перехода, всегда полезной?

I ₁	096	ADD
I ₂	100	BEQ x1,x2 +200
I ₃	104	ADD
I ₄	300	ADD

Branch Delay Slots

(сделать коллизии по управлению видимыми ПО)

- Поменять семантику АНК так, чтобы инструкция, следующая за условным (branch) или безусловным (跳跃) переходом, всегда исполнялась
 - Предоставляет компилятору гибкость в виде возможности поместить полезную инструкцию туда, где иначе был бы конвейерный «пузырек»

I ₁	096	ADD
I ₂	100	BEQZ r1, +200
I ₃	104	ADD
I ₄	300	ADD

Инструкция в *delay slot*'е
исполняется независимо
от результатов
вычисления перехода

Диаграмма конвейера с условным переходом (с использованием branch delay slot)

	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7
(I ₁) 096: ADD	IF ₁	ID ₁	EX ₁	MA ₁	WB ₁				
(I ₂) 100: BEQZ +200		IF ₂	ID ₂	EX ₂	MA ₂	WB ₂			
(I ₃) 104: ADD			IF ₃	ID ₃	EX ₃	MA ₃	WB ₃		
(I ₄) 300: ADD				IF ₄	ID ₄	EX ₄	MA ₄	WB ₄	

<i>Resource Usage</i>	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7
	IF	I ₁	I ₂	I ₃	I ₄				
	ID		I ₁	I ₂	I ₃	I ₄			
	EX			I ₁	I ₂	I ₃	I ₄		
	MA				I ₁	I ₂	I ₃	I ₄	
	WB					I ₁	I ₂	I ₃	I ₄

АНК после 90-х не имеют branch delay slot'ов

- Вводит микроархитектурный элемент в АНК
- Какие проблемы есть с delay slot'ами?

АНК после 90-х не имеют branch delay slot'ов

- Вводит микроархитектурный элемент в АНК
- Какие проблемы есть с delay slot'ами?
- Проблемы производительности
 - Например, промах в кэш инструкций или page fault в delay slot'e потребуют от машины ожидания, даже если инструкция в delay slot'e была NOP
- Усложняет более продвинутые архитектуры
 - Конвейер из 30 стадий с 4 подаваемыми на вход конвейера инструкциями за такт
- Усложняет работу компилятора
- Более точное предсказание переходов уменьшило необходимость в delay slot'ах

Почему инструкцию нельзя отправлять на конвейер на каждом такте ($CPI > 1$)

Почему инструкцию нельзя отправлять на конвейер на каждом такте ($CPI > 1$)

- Полное байпасирование может оказаться слишком дорогим для имплементации
 - Обычно реализуются все часто используемые пути
 - Некоторые редко используемые пути байпаса могут увеличить длительность такта, нивелируя тем самым пользу от уменьшения CPI

Почему инструкцию нельзя отправлять на конвейер на каждом такте ($CPI > 1$)

- Полное байпасирование может оказаться слишком дорогим для имплементации
 - Обычно реализуются все часто используемые пути
 - Некоторые редко используемые пути байпasa могут увеличить длительность такта, нивелируя тем самым пользу от уменьшения CPI
- Load'ы имеют двухтактовую длительность
 - Инструкция после инструкции load не может использовать ее результат
 - АНК MIPS-I использовала load delay slot'ы для предотвращения коллизий, связанных с этой проблемой. Компилятор планировал независимую инструкцию или вставлял NOP. Load delay slot'ы были убраны в MIPS-II (в аппаратуру была добавлена возможность возникновения конвейерных interlock'ов)

Почему инструкцию нельзя отправлять на конвейер на каждом такте ($CPI > 1$)

- Полное байпасирование может оказаться слишком дорогим для имплементации
 - Обычно реализуются все часто используемые пути
 - Некоторые редко используемые пути байпasa могут увеличить длительность такта, нивелируя тем самым пользу от уменьшения CPI
- Load'ы имеют двухтактовую длительность
 - Инструкция после инструкции load не может использовать ее результат
 - АНК MIPS-I использовала load delay slot'ы для предотвращения коллизий, связанных с этой проблемой. Компилятор планировал независимую инструкцию или вставлял NOP. Load delay slot'ы были убраны в MIPS-II (в аппаратуру была добавлена возможность возникновения конвейерных interlock'ов)
- Условные переходы могут вызывать «пузырьки» в конвейере
 - «Убить» следующую инструкцию, если нет delay slot'ов

Почему инструкцию нельзя отправлять на конвейер на каждом такте ($CPI > 1$)

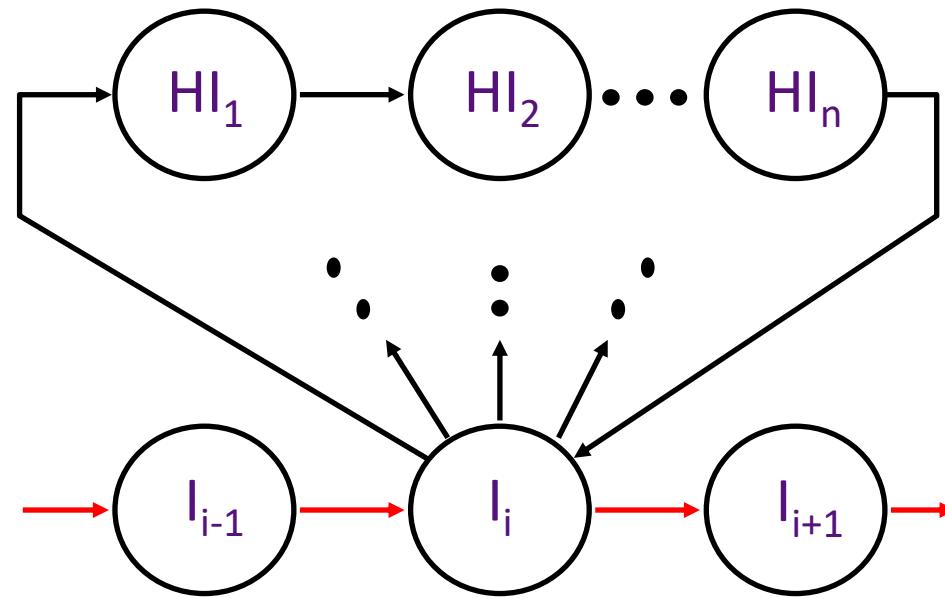
- Полное байпасирование может оказаться слишком дорогим для имплементации
 - Обычно реализуются все часто используемые пути
 - Некоторые редко используемые пути байпаса могут увеличить длительность такта, нивелируя тем самым пользу от уменьшения CPI
- Load'ы имеют двухтактовую длительность
 - Инструкция после инструкции load не может использовать ее результат
 - АНК MIPS-I использовала load delay slot'ы для предотвращения коллизий, связанных с этой проблемой. Компилятор планировал независимую инструкцию или вставлял NOP. Load delay slot'ы были убраны в MIPS-II (в аппаратуру была добавлена возможность возникновения конвейерных interlock'ов)
- Условные переходы могут вызывать «пузырьки» в конвейере
 - «Убить» следующую инструкцию, если нет delay slot'ов

Машины с программно видимыми delay slot'ами могут выполнять значительное число NOP-инструкций, вставляемых компилятором. Такие NOPы увеличивают число инструкций и размер программы!

Прерывания

Interrupt Handler

Программа



- Прерывание (англ. *interrupt*) — сигнал, сообщающий процессору о наступлении какого-либо события. При этом выполнение текущей последовательности команд приостанавливается, и управление передаётся обработчику прерывания (*Interrupt Handler*), который реагирует на событие и обслуживает его, после чего возвращает управление в прерванный код.

Типы прерываний

- **Асинхронные или внешние (аппаратные)** — события, которые исходят от внешних источников и могут произойти в любой произвольный момент. Факт возникновения в системе такого прерывания трактуется как запрос на прерывание (англ. Interrupt request, IRQ)
 - I/O Device service-request, таймер, hardware/power failure
- **Синхронные или внутренние (traps и exceptions)** — события в самом процессоре как результат нарушения каких-то условий при исполнении машинного кода
 - Undefined opcode, arithmetic overflow, FPU exception, misaligned memory
 - Virtual memory exceptions: page faults, TLB misses, protection violations
 - Traps (программные) — инициируются исполнением специальной инструкции в коде программы (system calls)

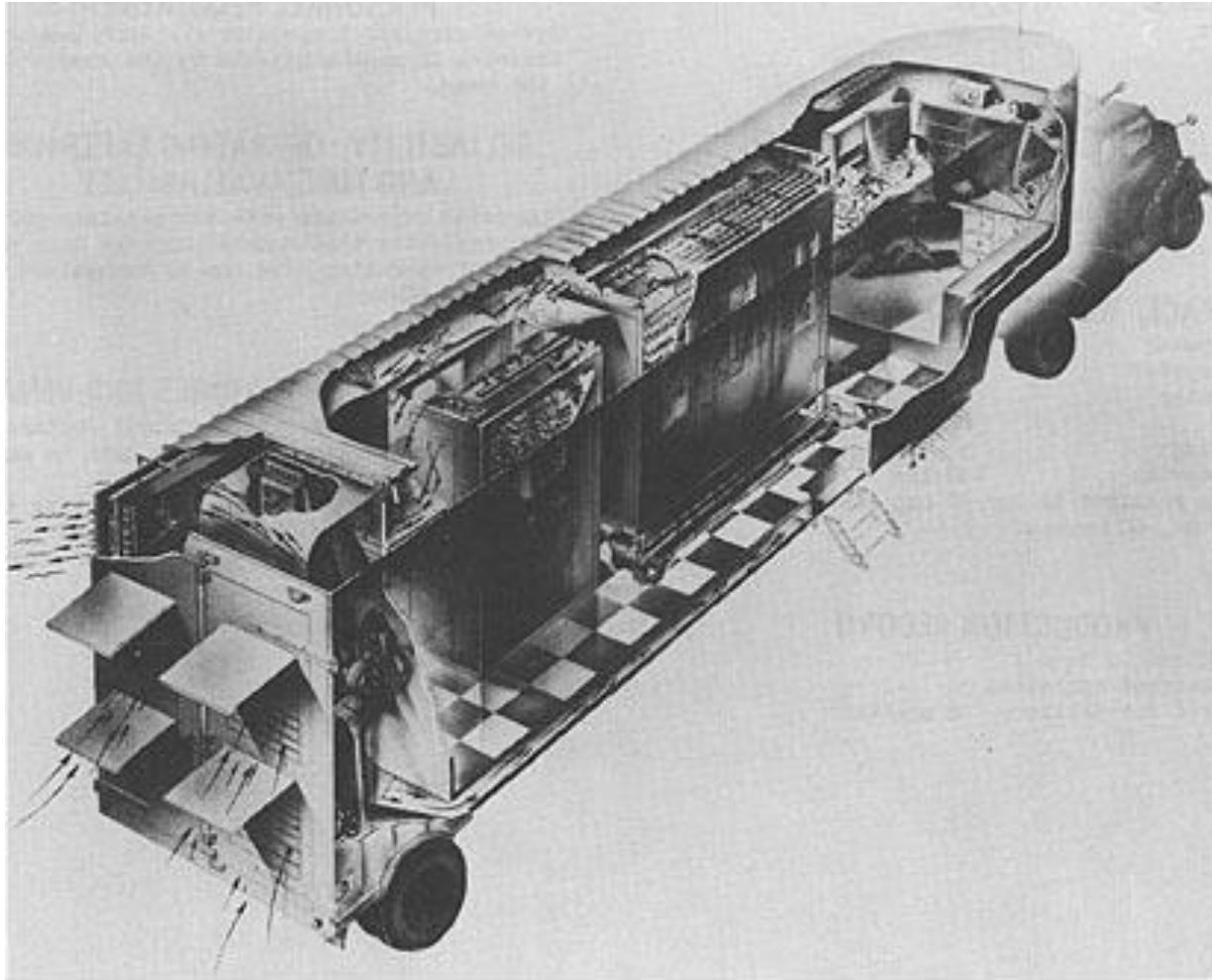
Типы прерываний

- **Маскируемые** — прерывания, которые можно запрещать установкой соответствующих битов в регистре маскирования прерываний (в x86-процессорах — сбросом флага IF в регистре флагов)
- **Немаскируемые** (англ. Non maskable interrupt, NMI) — обрабатываются всегда, независимо от запретов на другие прерывания

Первые системы с обработкой прерываний

- Univac-I (1951) – первая система с обработкой прерываний
 - Arithmetic overflow
 - Позднее в Univac 1103 (1955) появилась поддержка внешних прерываний
- DYSEAC (1954) – первая система с обработкой прерываний от ввода-вывода
 - Два указателя РС, переключение между ними по сигналу от ввода-вывода
 - Первая система с DMA (Direct Memory Access by I/O device)
 - Первый мобильный компьютер

DYSEAC, первый мобильный компьютер



- Две фуры, 12 тонн + 8 тонн

Точные прерывания

Прерывание, при обработке которого машина остается во вполне определенном состоянии, называется **точным** прерыванием (Уолкер (Walker) и Крэгон (Cragon), 1995).

- Счетчик команд сохранен в определенном месте
- Все команды, предшествующие той, на которую указывает счетчик команд, были полностью выполнены
- Ни одна из команд, следующих за той, на которую указывает счетчик команд, не была выполнена
- Известно состояние выполнения той команды, на которую указывает счетчик команд

Обработка прерываний

- Текущая программа остановлена на команде I_i , все предыдущие команды завершены до I_{i-1} включительно (*точное прерывание*)
- Значение РС команды I_i сохранено в специальный регистр (EPC)
- Новые прерывания замаскированы, управление передается обработчику соответствующего прерывания (Interrupt Handler), исполняющемуся в kernel mode

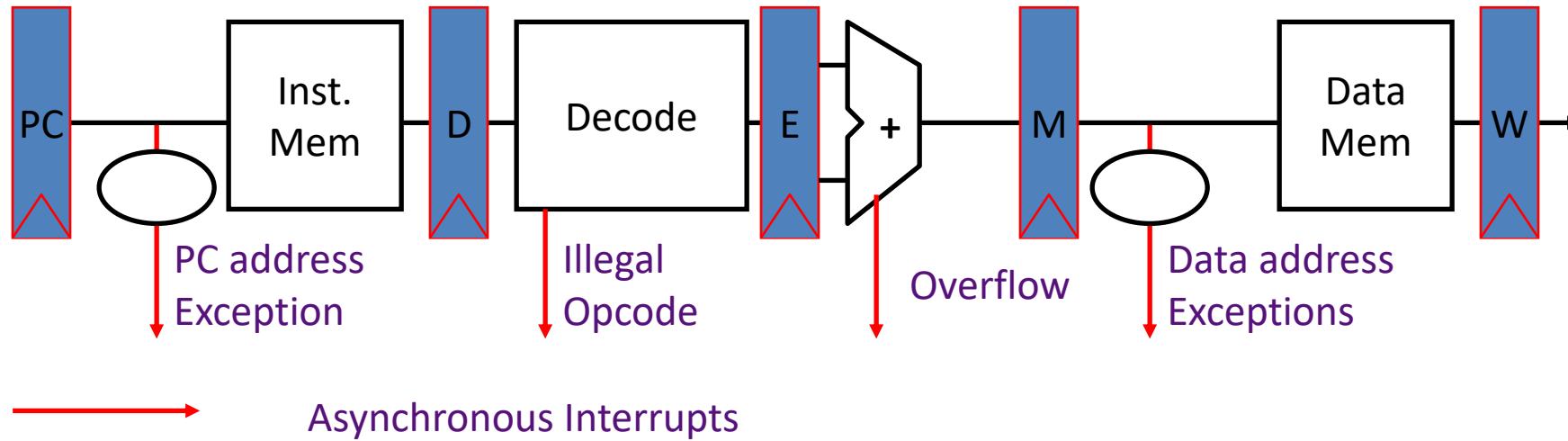
Обработчик прерываний

- Сохранение ЕРС до включения прерываний (отложенные прерывания) ⇒
 - команда записи ЕРС в GPRs
 - до завершения записи ЕРС новые прерывания должны быть замаскированы
- Чтение регистра статуса (Cause register) для определения причины прерывания
- Специальная команда перехода SRET (*return-from-supervisor*)
 - включение прерываний
 - возвращение в пользовательский режим
 - восстановление архитектурного состояния

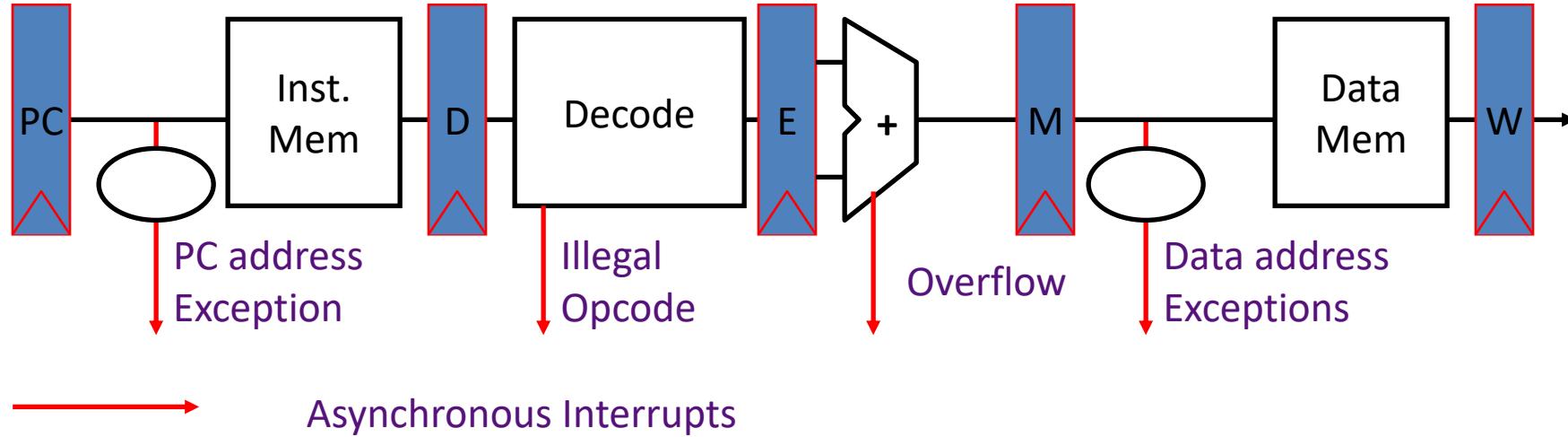
Синхронное прерывание

- Возникает во время исполнения *определенной команды*
- В общем случае команда не может быть завершена и должна быть повторно исполнена после обработки прерывания
- В случае системных вызовов команда (*system call*) считается исполненной
 - специальная команда перехода в *privileged kernel mode*

Обработка прерываний 5-Stage Pipeline

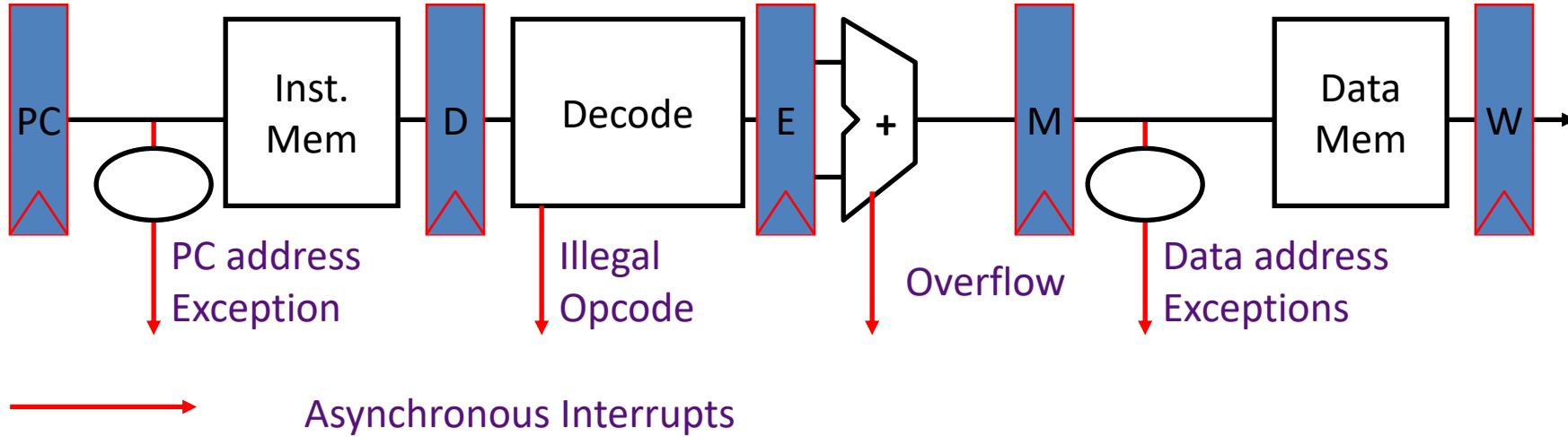


Обработка прерываний 5-Stage Pipeline



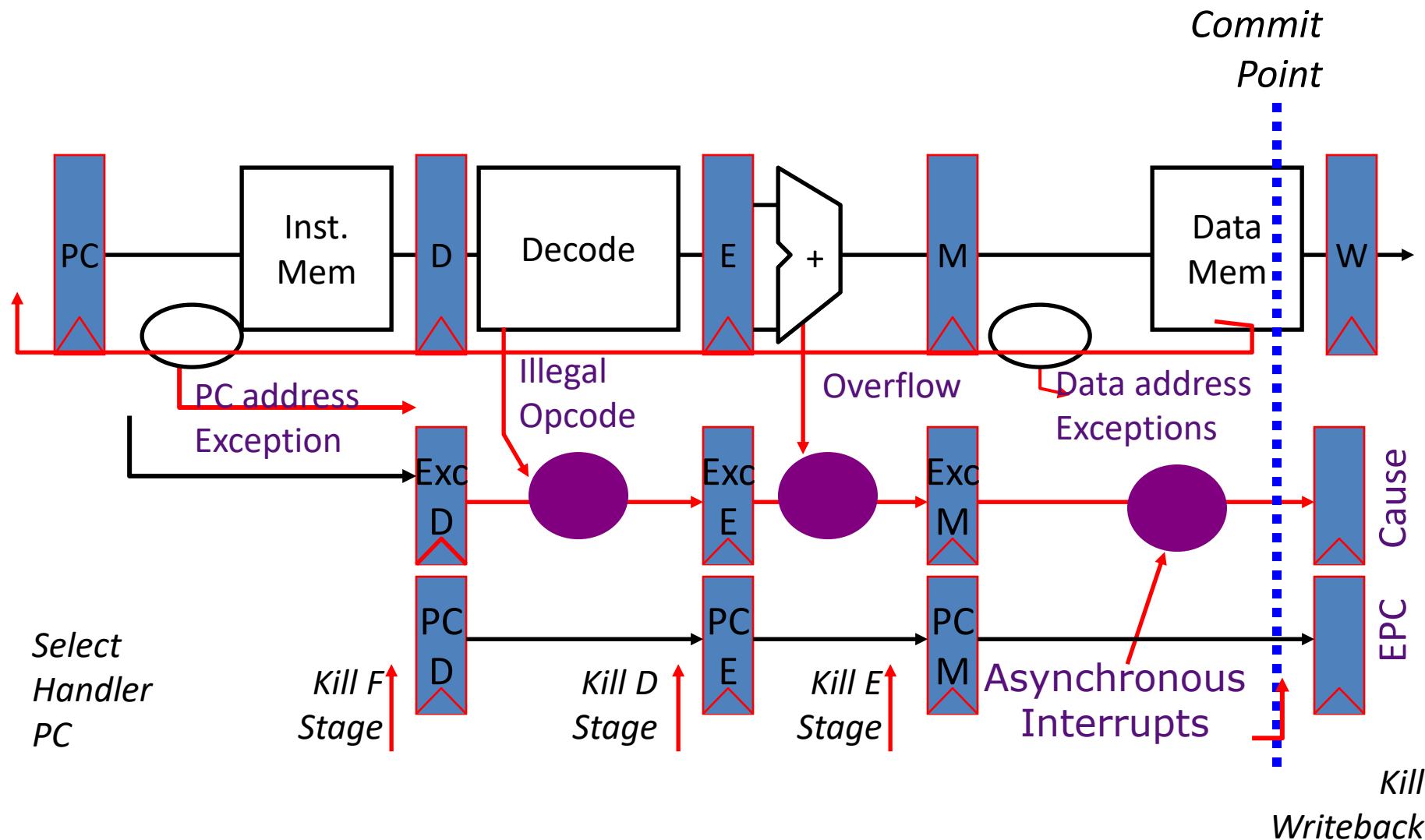
- Как обработать несколько синхронных прерываний с разных стадий конвейера?

Обработка прерываний 5-Stage Pipeline



- Как обработать несколько синхронных прерываний с разных стадий конвейера?
- Как и где обработать внешние асинхронные прерывания?

Обработка прерываний 5-Stage Pipeline



Обработка прерываний 5-Stage Pipeline

- Передавать признаки синхронных прерываний (exception flags) по конвейеру до commit point (M stage)
- Для каждой команды прерывание на ранних стадиях более приоритетно
- Асинхронное прерывание имеет наибольший приоритет (commit point)
- Если есть признак прерывания в commit point:
 - обновить регистры Cause and EPC
 - удалить команды со всех стадий конвейера
 - перейти к исполнению обработчика прерываний (handler PC)

Спекулятивность и прерывания

- Механизм предсказания прерываний
 - Прерывания очень редки, поэтому предсказание, что прерывания не будет, очень точно!
- Механизм проверки предсказания прерываний
 - На последней стадии конвейера
- Механизм восстановления
 - Обновление архитектурного состояния только в commit point, поэтому команды на всех предыдущих стадиях конвейера можно просто удалить
 - После очистки конвейера установить interrupt handler PC
- Байпассирование позволяет использовать результаты команд, еще не прошедших commit point

Диаграмма конвейера при прерывании

	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7	...
(I ₁) 096: ADD	IF ₁	ID ₁	EX ₁	MA ₁	-				<i>overflow!</i>
(I ₂) 100: XOR		IF ₂	ID ₂	EX ₂	-				
(I ₃) 104: SUB			IF ₃	ID ₃	-				
(I ₄) 108: ADD				IF ₄	-				
(I ₅) Exc. Handler code					IF ₅	ID ₅	EX ₅	MA ₅	WB ₅

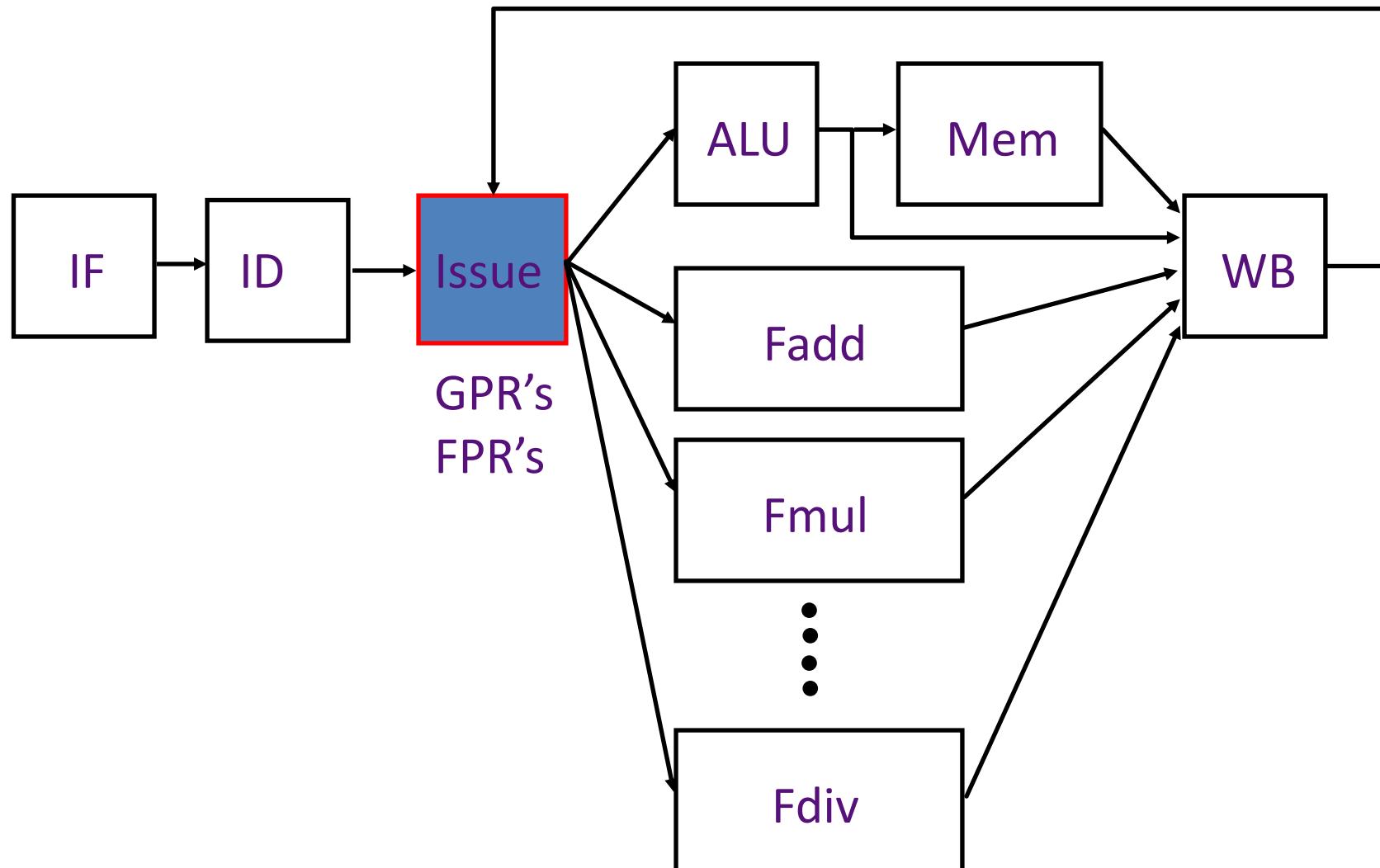
<i>Resource Usage</i>	<i>time</i>								
	t0	t1	t2	t3	t4	t5	t6	t7	...
	IF	I ₁	I ₂	I ₃	I ₄	I ₅			
	ID		I ₁	I ₂	I ₃	-	I ₅		
	EX			I ₁	I ₂	-	-	I ₅	
	MA				I ₁	-	-	-	I ₅
	WB					-	-	-	I ₅

Увеличение количества стадий

Предпосылки:

- Большие задержки доступа к памяти
- Большое время работы FPU
- Не конвейеризованные исполнительные устройства
- Наличие большого количества разнообразных устройств с разным временем работы

Новая структура конвейера



Новые коллизии

- Структурный конфликт на стадии WB при записи в память данных из разных устройств
- Структурный конфликт, так как некоторые FPU и устройства подсистемы памяти не конвейеризируемые и/или работают дольше одного такта
- Коллизии по управлению при обработке прерываний

Следующая лекция

- Динамическое планирование
 - Схема Scoreboarding
 - Внеочередное исполнение команд
 - Переименование регистров и алгоритм Томасуло
 - Reorder buffer
-
- Материалы для сегодняшней лекции
 - H&P Appendix C.1-C.4, 3.1
 - Материалы для следующей лекции
 - H&P Appendix C.5-C.8, 3.4-3.6

Благодарности

- Курс основан на материалах курсов по архитектуре микропроцессоров следующих авторов:
 - Arvind (MIT)
 - Joel Emer (Intel/MIT)
 - James Hoe (CMU)
 - John Kubiatowicz (UCB)
 - David Patterson (UCB)
 - Krste Asanovic (UCB)
 - David Wentzlaff (Princeton)