Тема 2. Организация компьютерных систем

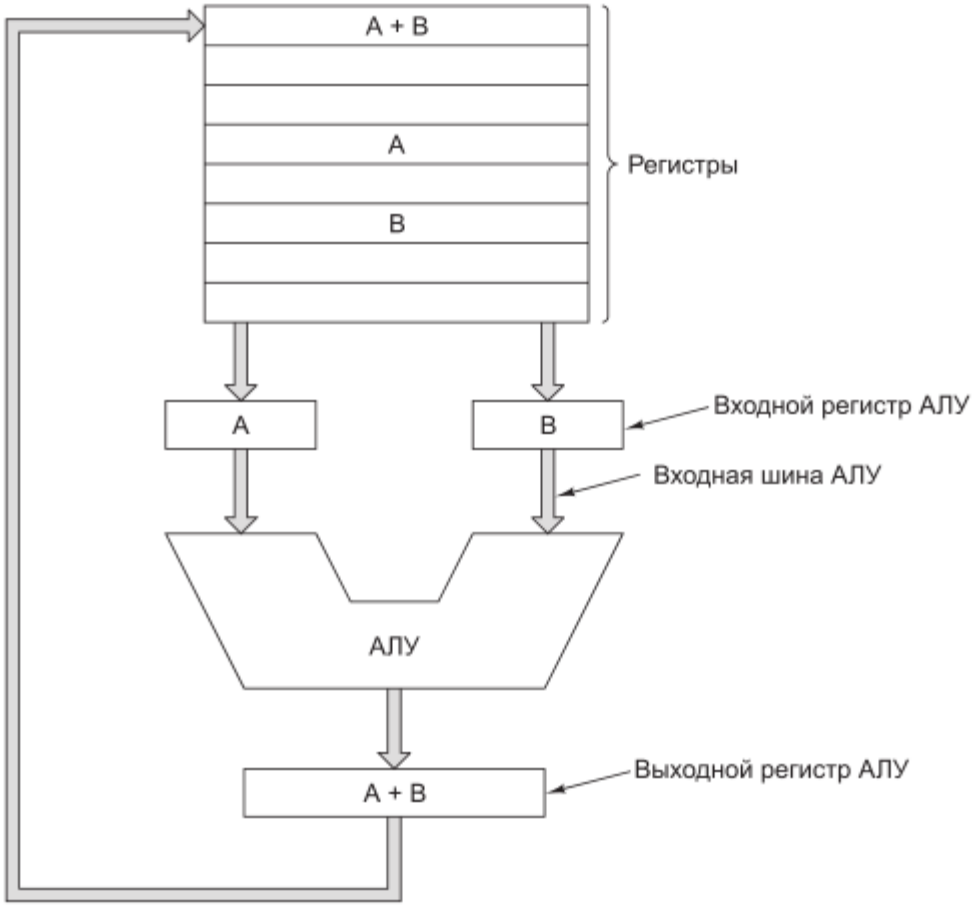
Организация конвейера.

Конфликты конвейерных систем.

Устройство ЦП

Выполнение команд (выборка – декодирование – выполнение)

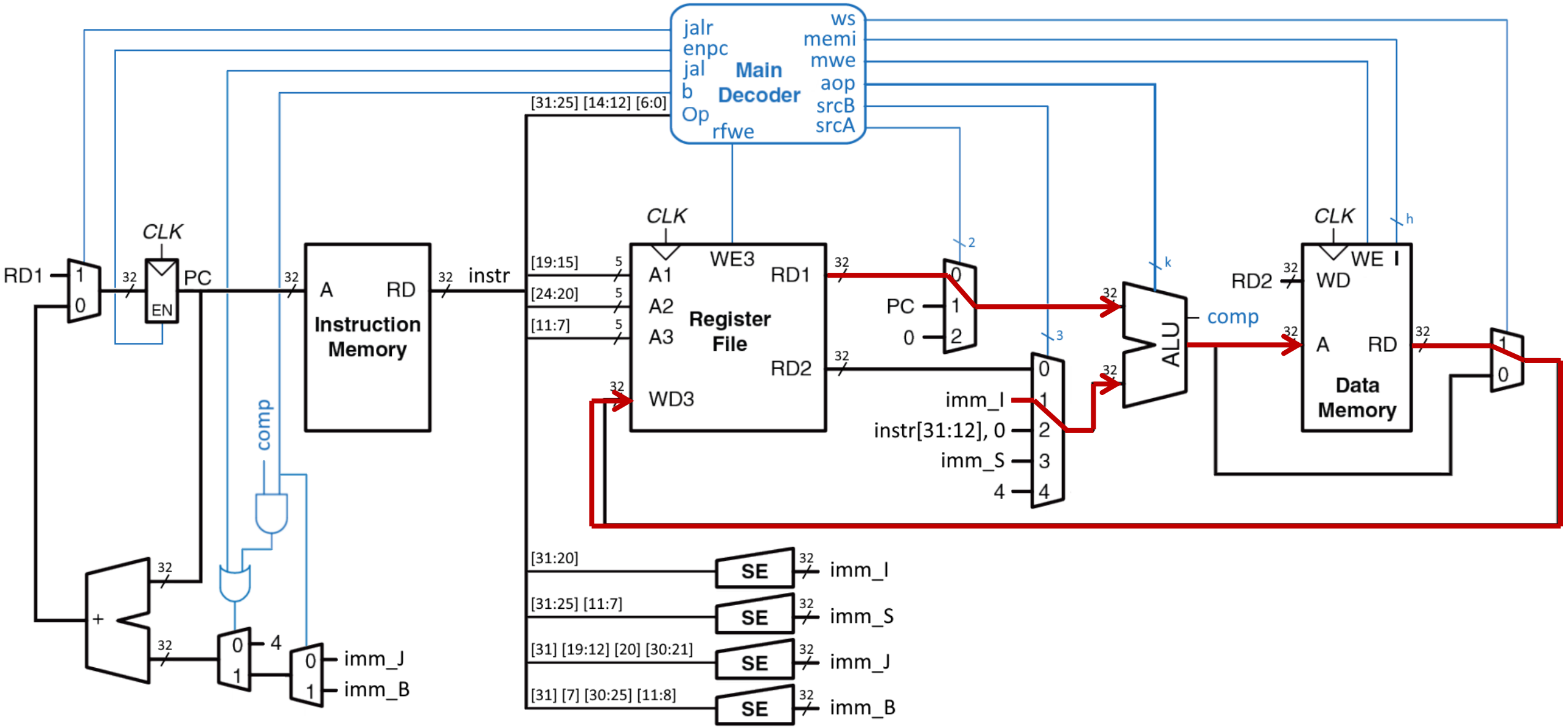
1. Вызывает следующую команду из памяти и переносит ее в регистр команд.
2. Меняет положение счетчика команд, который после этого указывает на следующую команду
3. Определяет тип вызванной команды.
4. Если команда использует слово из памяти, определяет, где находится это слово.
5. Переносит слово, если это необходимо, в регистр центрального процессора
6. Выполняет команду.
7. Переходит к шагу 1, чтобы начать выполнение следующей команды.



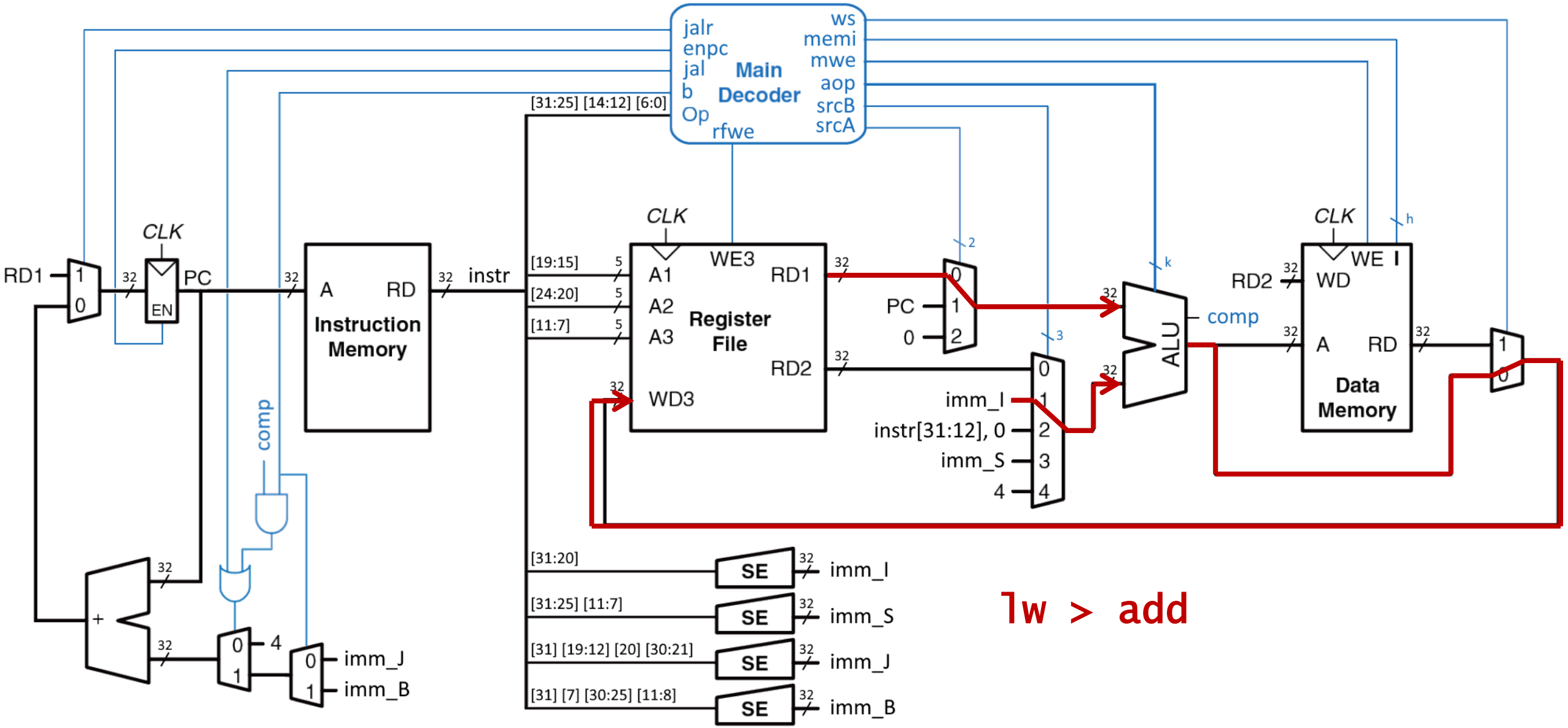
Микроархитектуры

- **Однотактная**
  - выполняет одну инструкцию за один такт

lw


jalr
enpc
jal
b
Op
Main Decoder
rfwe
ws
memi
mwe
aop
srcB
srcA
[31:25] [14:12] [6:0]
CLK
RD1
1
0
32
EN
PC
32
A
RD
32
instr
Instruction Memory
[19:15]
[24:20]
[11:7]
5
5
5
A1
A2
A3
WD3
WE3
RD1
RD2
Register File
32
32
32
2
PC
0
1
2
0
32
3
imm_I
instr[31:12], 0
imm_S
4
0
1
2
3
4
k
32
32
ALU
comp
CLK
h
RD2
32
WE
WD
32
A
RD
32
Data Memory
1
0
comp
32
+
32
32
0
1
4
0
1
imm_J
imm_B
[31:20]
[31:25] [11:7]
[31] [19:12] [20] [30:21]
[31] [7] [30:25] [11:8]
SE
SE
SE
SE
32
32
32
32
imm_I
imm_S
imm_J
imm_B

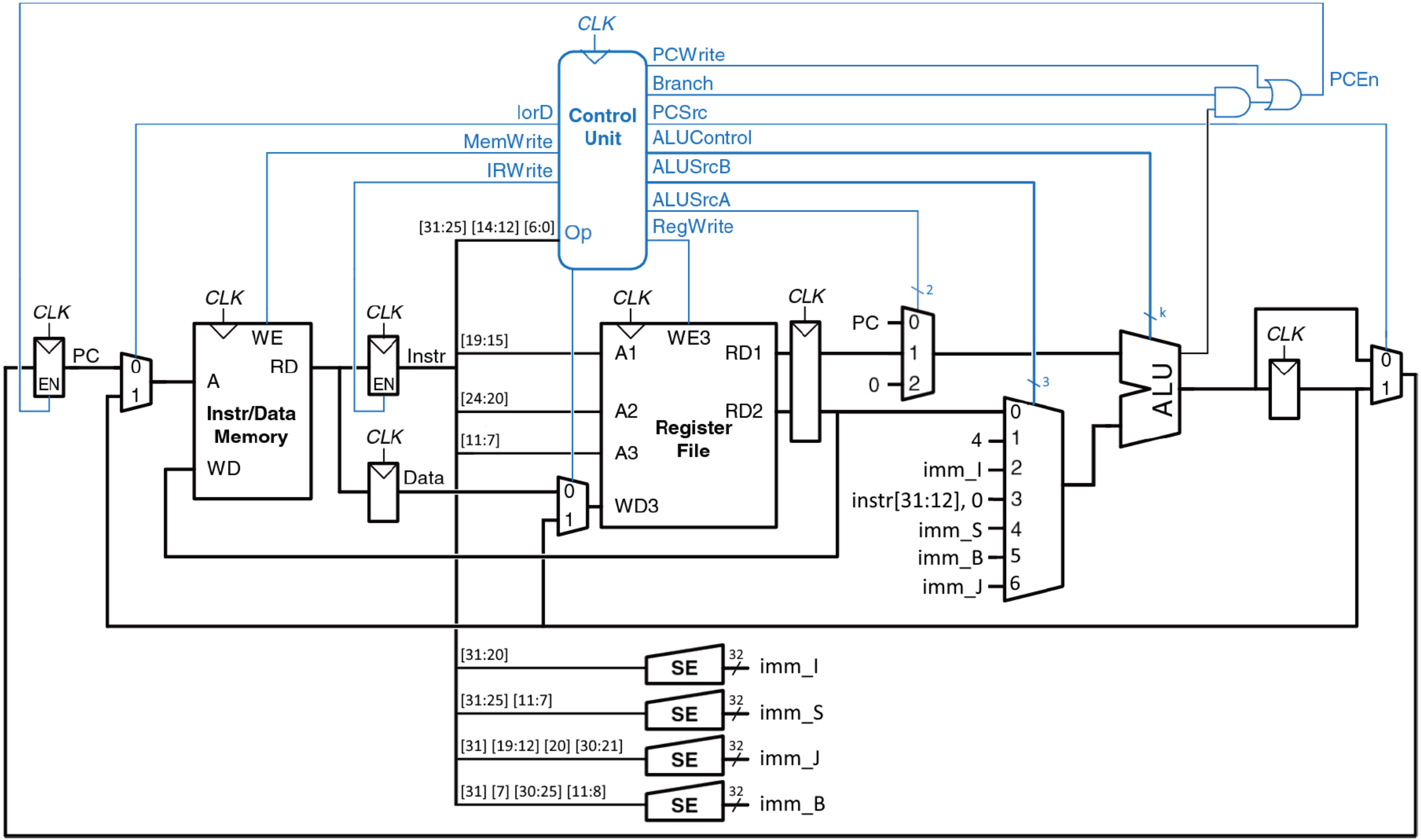
add


jalr
enpc
jal
b
Op
Main Decoder
rfwe
ws
memi
mwe
aop
srcB
srcA
[31:25] [14:12] [6:0]
CLK
RD1
1
0
32
EN
PC
32
A
RD
32
instr
Instruction Memory
[19:15]
[24:20]
[11:7]
5
5
5
A1
A2
A3
WD3
WE3
RD1
RD2
Register File
32
32
32
2
PC
0
1
2
0
imm_I
instr[31:12], 0
imm_S
4
1
2
3
4
3
k
ALU
comp
32
32
32
CLK
h
RD2
32
WD
WE
32
A
RD
32
Data Memory
1
0
comp
+
32
32
0
1
4
0
1
imm_J
imm_B
[31:20]
[31:25] [11:7]
[31] [19:12] [20] [30:21]
[31] [7] [30:25] [11:8]
SE
SE
SE
SE
32
32
32
32
imm_I
imm_S
imm_J
imm_B

lw > add

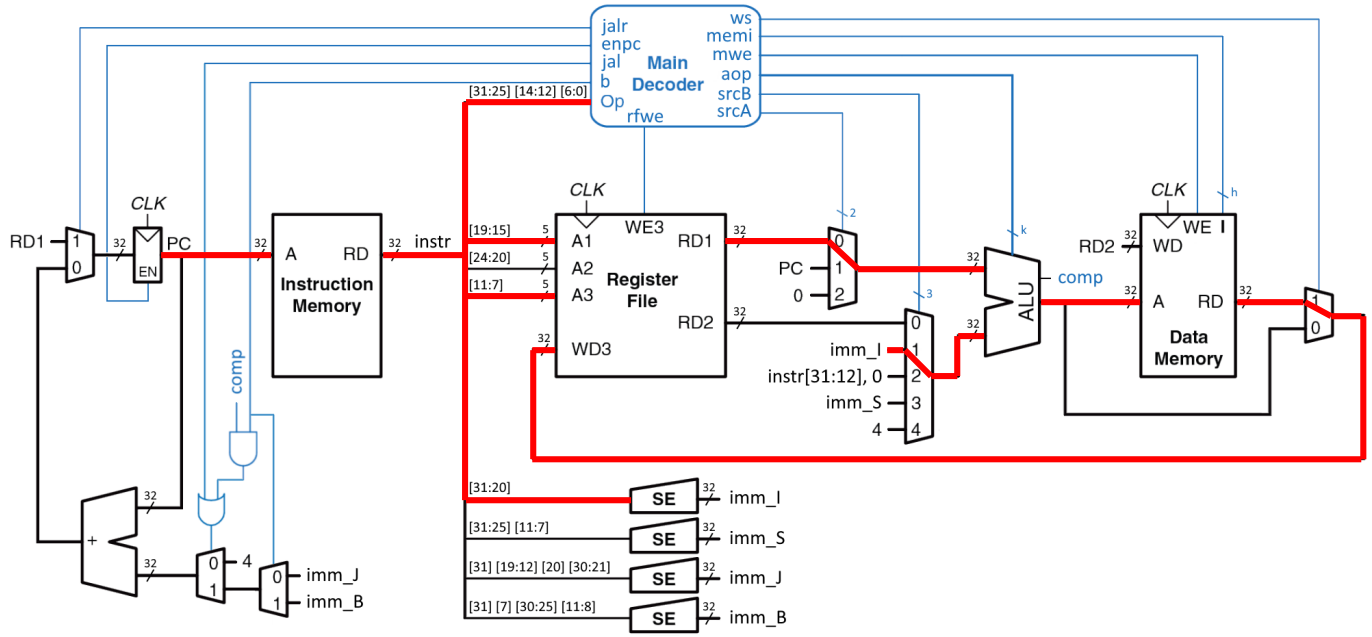
Микроархитектуры

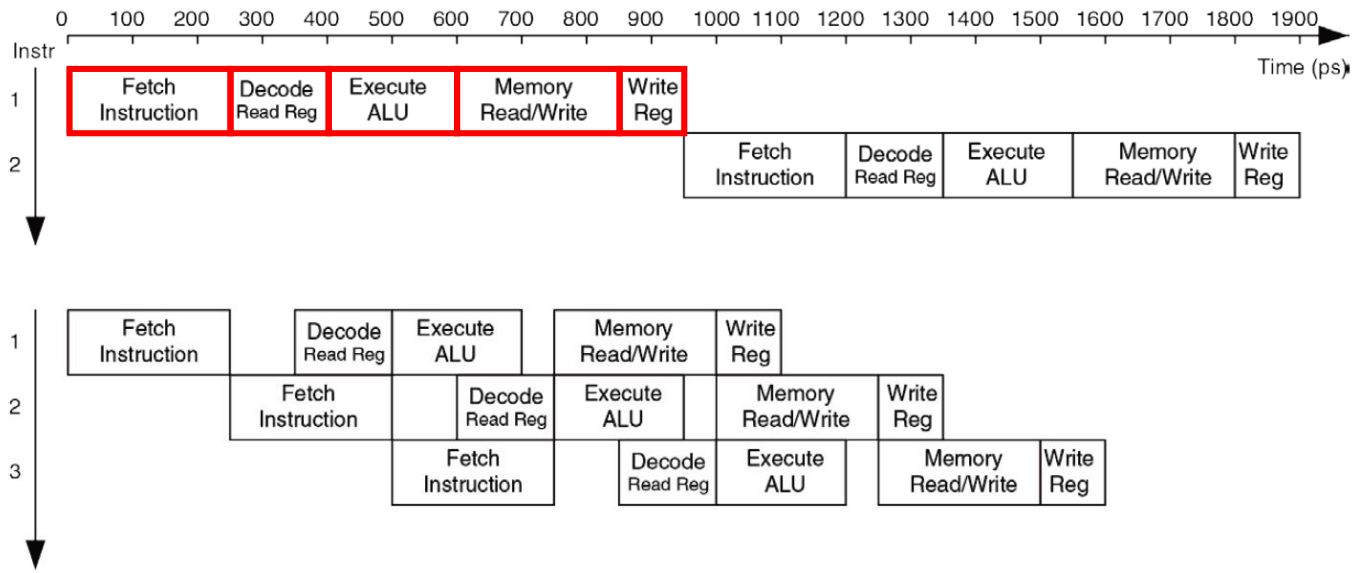
- **Однотактная**
  - выполняет одну инструкцию за один такт
- **Многотактная**
  - выполняет одну инструкцию за несколько более коротких тактов


CLK
PCWrite
Branch
PCSrc
ALUControl
ALUSrcB
ALUSrcA
RegWrite
Op
Control Unit
IorD
MemWrite
IRWrite
PCEn
[31:25] [14:12] [6:0]
CLK
PC
EN
0
1
WE
RD
A
Instr/Data Memory
WD
Instr
Data
[19:15]
[24:20]
[11:7]
A1
A2
A3
WE3
WD3
RD1
RD2
Register File
2
PC
0
1
2
k
3
4
imm_I
instr[31:12], 0
imm_S
imm_B
imm_J
3
4
5
6
ALU
[31:20]
[31:25] [11:7]
[31] [19:12] [20] [30:21]
[31] [7] [30:25] [11:8]
SE
32
imm_I
imm_S
imm_J
imm_B

Микроархитектуры

- **Однотактная**
  - выполняет одну инструкцию за один такт
- **Многотактная**
  - выполняет одну инструкцию за несколько более коротких тактов
- **Конвейерная**
  - результат применения принципа конвейерной обработки к однотактной микроархитектуре


jalr
enpc
jal
b
Op
Main Decoder
rfwe
ws
memi
mwe
aop
srcB
srcA
[31:25] [14:12] [6:0]
CLK
RD1
PC
EN
A
RD
Instruction Memory
instr
[19:15]
[24:20]
[11:7]
A1
A2
A3
WE3
WD3
RD1
RD2
Register File
PC
imm_I
instr[31:12], 0
imm_S
4
ALU
comp
RD2
WD
WE
Data Memory
imm_J
imm_B
[31:20]
[31:25] [11:7]
[31] [19:12] [20] [30:21]
[31] [7] [30:25] [11:8]
SE


0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600 1700 1800 1900
Time (ps)
Instr
1
2
Fetch Instruction
Decode Read Reg
Execute ALU
Memory Read/Write
Write Reg
3

Оценка производительности

Однотактный

T_1 = (100 × 10^9 команд) (1 такт/команду) (925 × 10^{-12} с/такт) = 92,5 с.

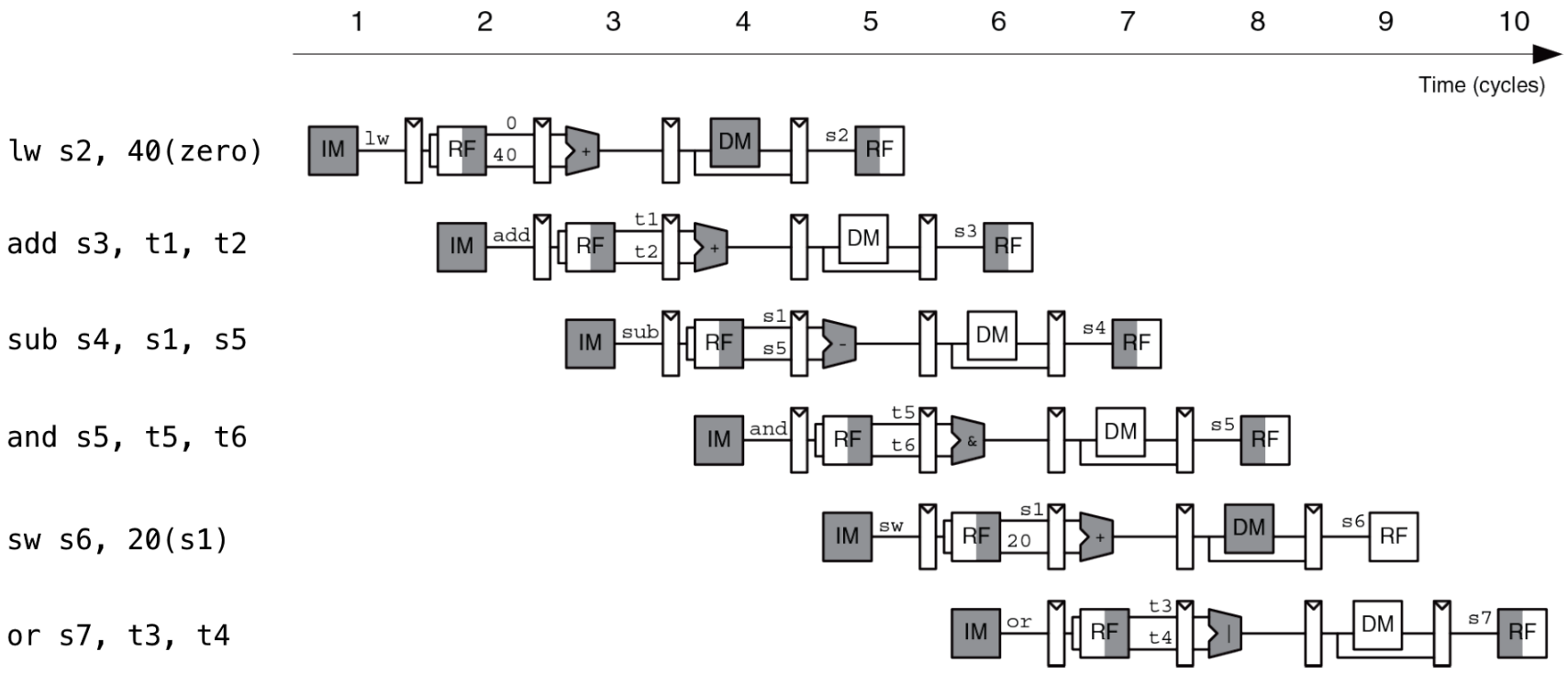
Многотактный

T_2 = (100 × 10^9 команд)(4,12 тактов/команду) (325 × 10^{-12} с/такт) = 133,9 с.

Конвейерный

T_3 = (100 × 10^9 команд)(1,15 тактов/команду)(550 × 10^{-12} с/такт) = 63,3 с.

Конвейер команд



Предпосылки параллелизма. Принципы RISC (недостижимый идеал)

- Все команды должны выполняться непосредственно аппаратным обеспечением
- Компьютер должен запускать как можно большее число операций в секунду (важен факт запуска)
- Команды должны легко декодироваться
- К памяти следует обращаться только командам загрузки и сохранения.
- Регистров должно быть много

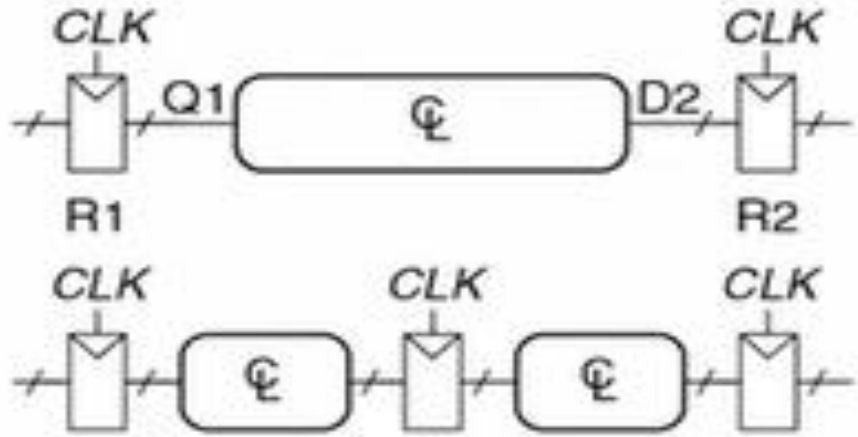
Параллелизм

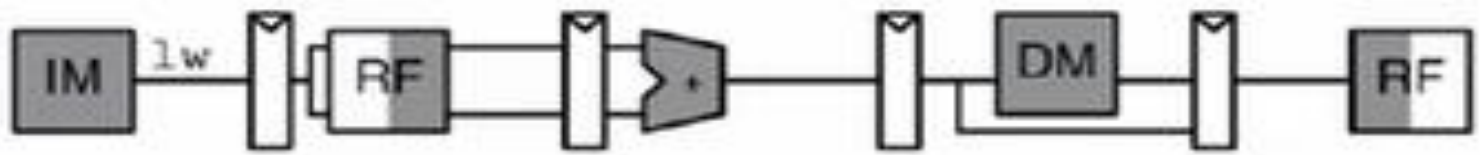
Выполнение двух или более команд одновременно

- Параллелизм на уровне команд
  - **Конвейеры**
  - **Суперскалярная архитектура**

- Параллелизм на уровне процессоров
  - Матричные компьютеры
  - Векторные компьютеры
  - Мультипроцессоры
  - Мультикомпьютеры

Конвейер

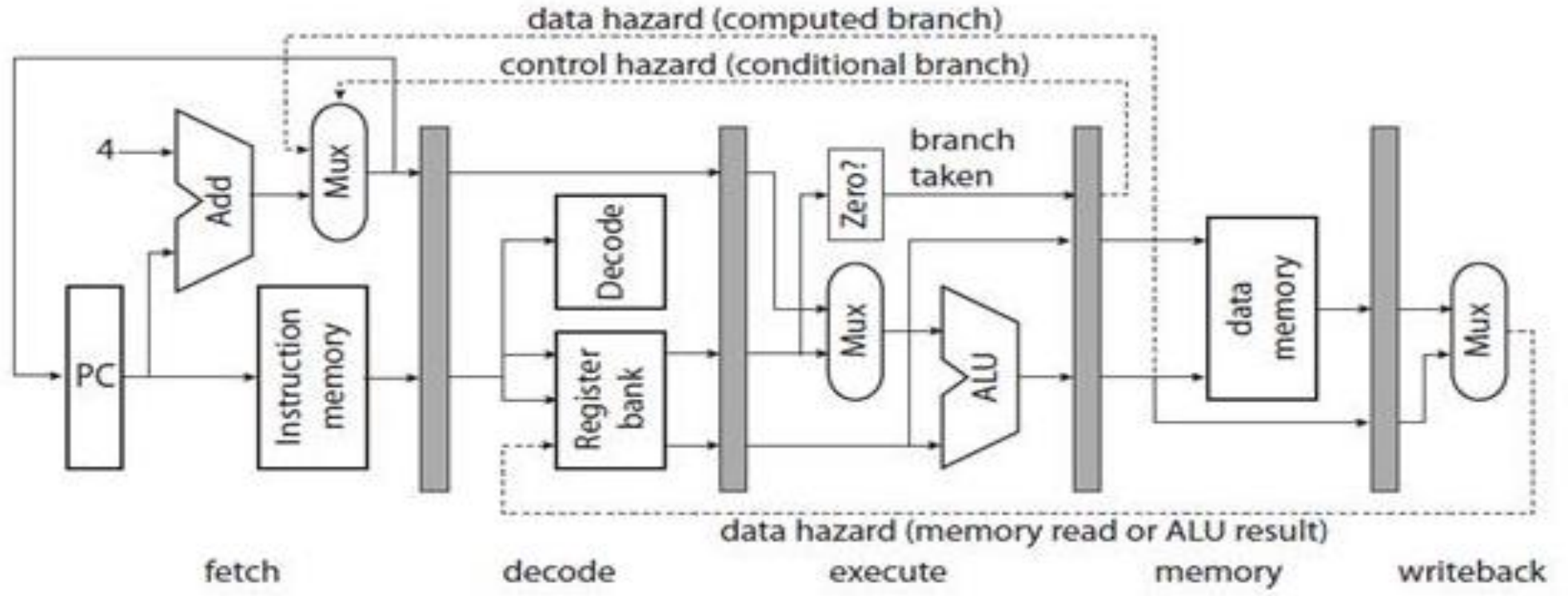




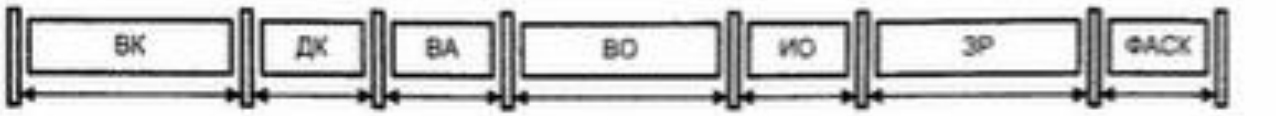
Fetch	Decode	Execute	Memory	Writeback

FET	DEC	ADDR	OP	EXE	WB

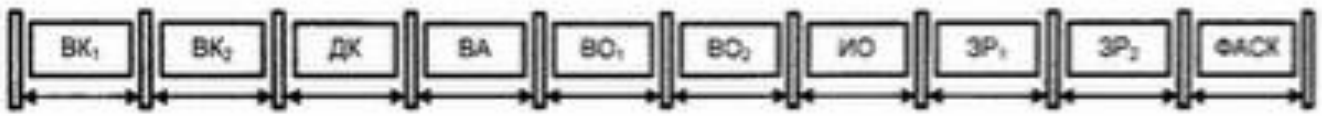
Пятиступенчатый конвейер для 32-разрядного процессора.



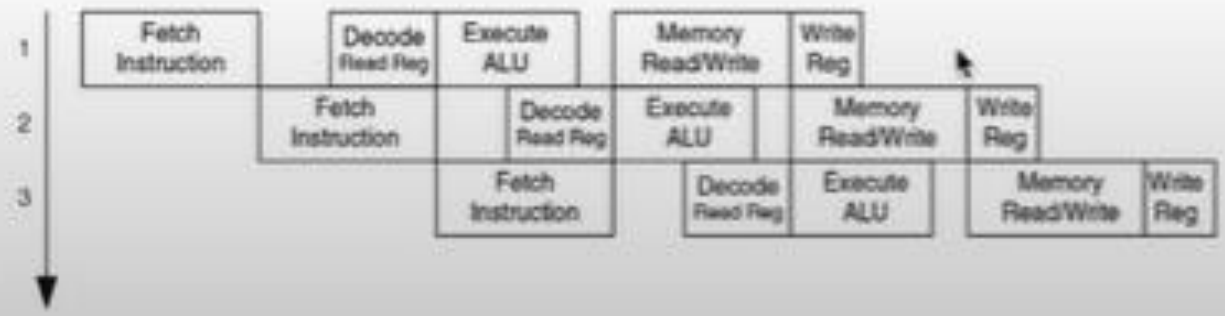
Суперконвейер

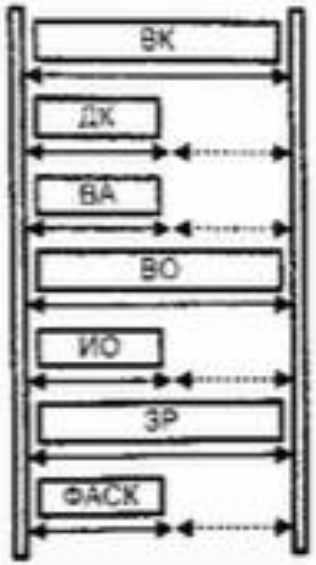


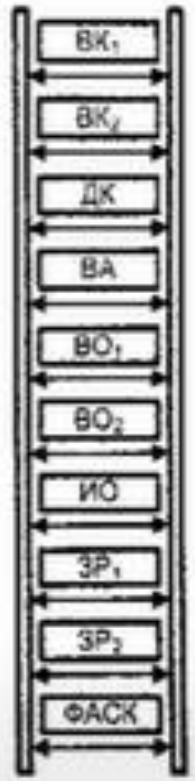
а



б







Конфликты конвейера

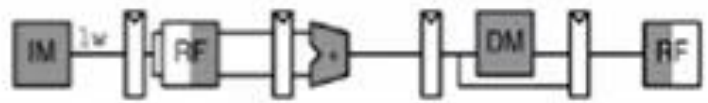
Структурные

```
mul s1, s1, s2
mul t2, t4, t2
mul t1, t3, t3
```

По данным

```
add s3, t1, t2
sub s4, s3, s1
or  s5, s3, t2
```

По управлению




```
(if or else)?
if(a < b)
```

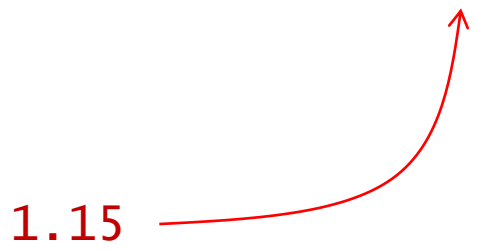
Производительность конвейера

- Состояние ожидания – конвейер пропускает один или несколько тактов из-за того, что не готовы операнды
- Состояние простоя – конвейер пропускает один или несколько тактов потому, что данный этап конвейера не используется в данной команде

$$CPI_{pipeline} = CPI_{ideal\_pipeline} + CPI_{SH} + CPI_{DH} + CPI_{CH}$$

Оценка производительности

$$Execution\ Time = (\#\ instructions)\left(\frac{cycles}{instruction}\right)\left(\frac{seconds}{cycle}\right)$$



Оценка производительности

Однотактный

T_{c1} = 30 + 2(250) + 150 + 200 + 25 + 20 = 925 пс.

T_1 = (100 × 10^9 команд) (1 такт/команду) (925 × 10^{-12} с/такт) = 92,5 с.

Многотактный

$T_c = t_{pcq} + t_{mux} + max(t_{ALU} + t_{mux}, t_{mem}) + t_{setup}$

T_{c2} = 30 + 25 + 250 + 20 = 325 пс.

T_2 = (100 × 10^9 команд)(4,12 тактов/команду) (325 × 10^{-12} с/такт) = 133,9 с.

Оценка производительности

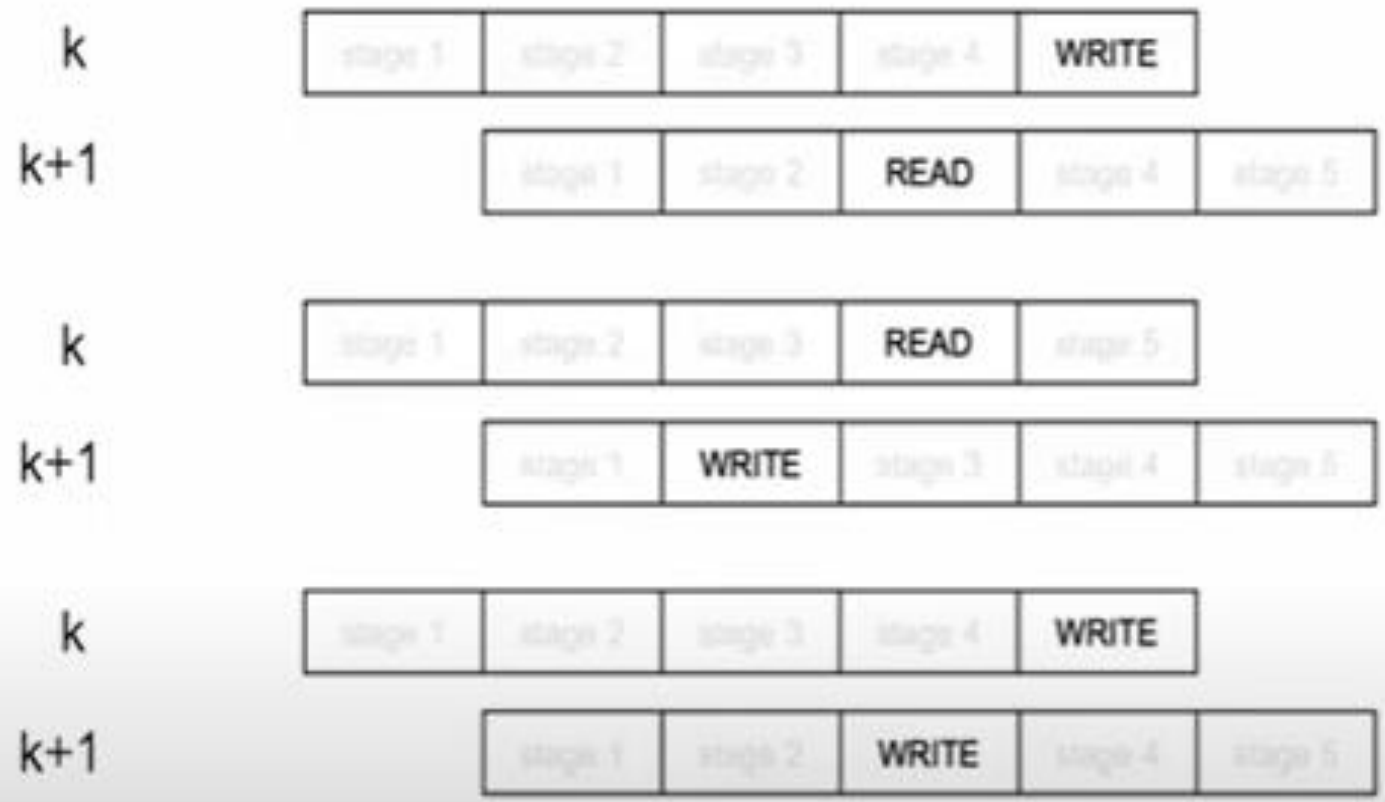
Конвейерный

$$T_c = max \left\{ \begin{array}{ll} t_{pcq}+t_{mem}+t_{setup} & \text{Fetch} \\ 2(t_{RFread}+t_{mux}+t_{eq}+t_{AND}+T_{mux}+t_{setup}) & \text{Decode} \\ t_{pcq}+t_{mux}+t_{mux}+t_{ALU}+t_{setup} & \text{Execute} \\ t_{pcq}+t_{memwrite}+t_{setup} & \text{Memory} \\ (t_{pcq}+t_{mux}+t_{RFwrite}) & \text{Writeback} \end{array} \right.$$

T_{c3} = 550 пс.

T_3 = (100 × 10^9 команд)(1,15 тактов/команду)(550 × 10^{-12} с/такт) = 63,3 с.

Конфликты по данным



read after write

RAW

write after read

WAR

write after write

WAW

Способы разрешения конфликтов.

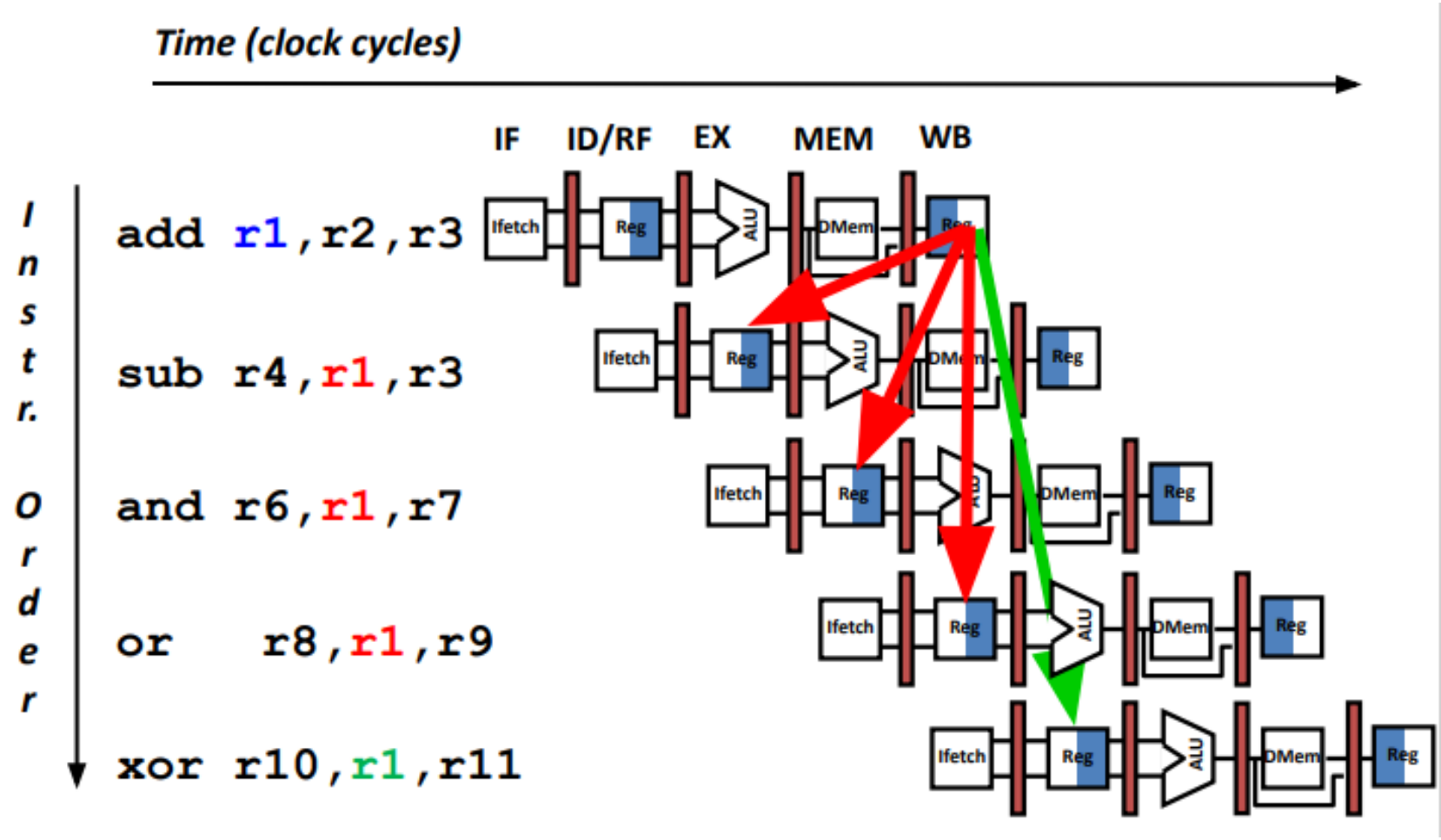
Программные

- Обнаружение конфликтов в процессе компиляции и добавление инструкций nop.
- Статическое планирование конвейера.

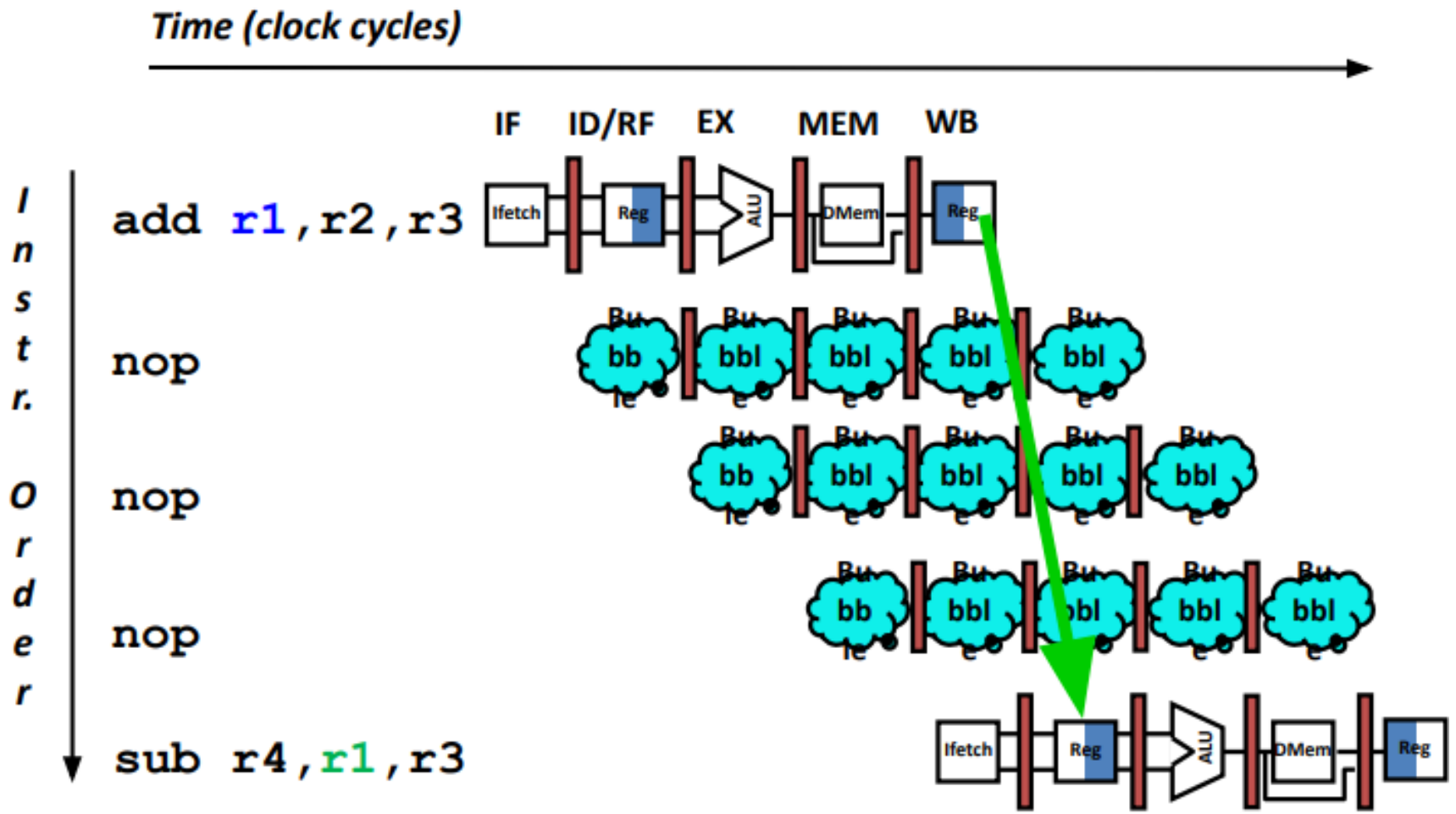
Аппаратные

- Добавление устройства обнаружения конфликтов и останова конвейера.
- Добавление специфичных аппаратных решений.

Конфликт RAW



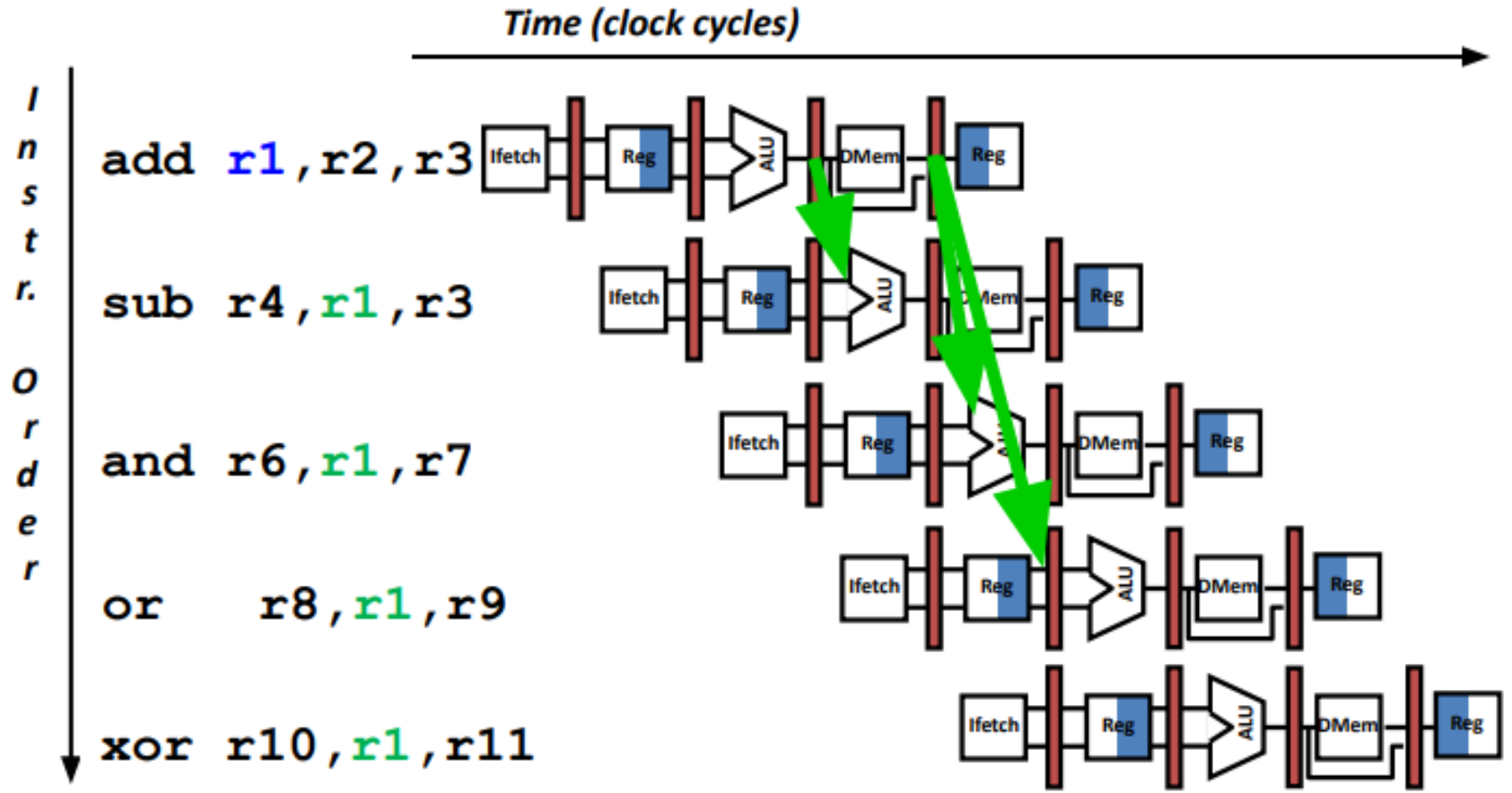
Конфликт RAW



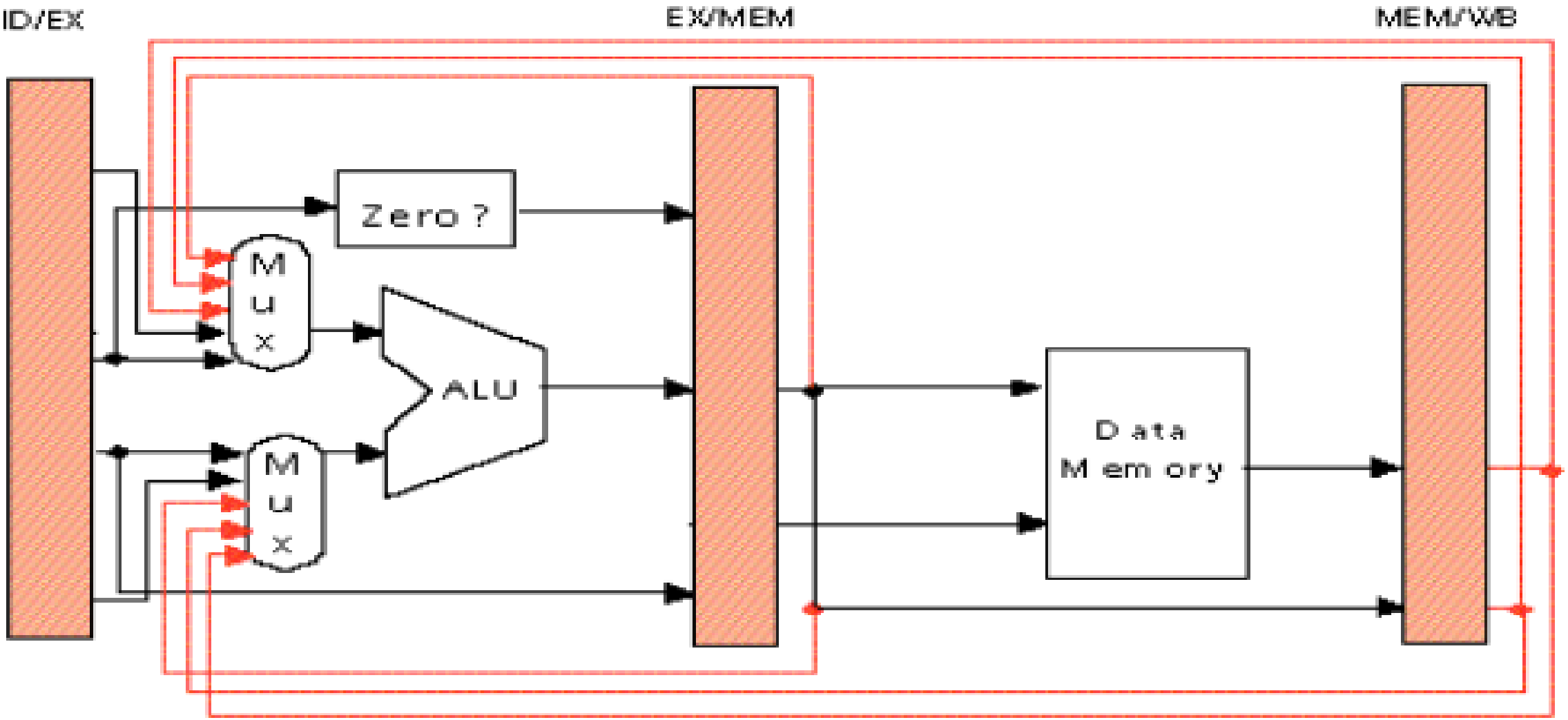
Устранение RAW конфликтов методом bypassing

- Метод Bypass сводиться к возможности передачи данных между стадиями конвейера напрямую.
- Вход стадии может быть соединен с выходом любой последующей стадии.

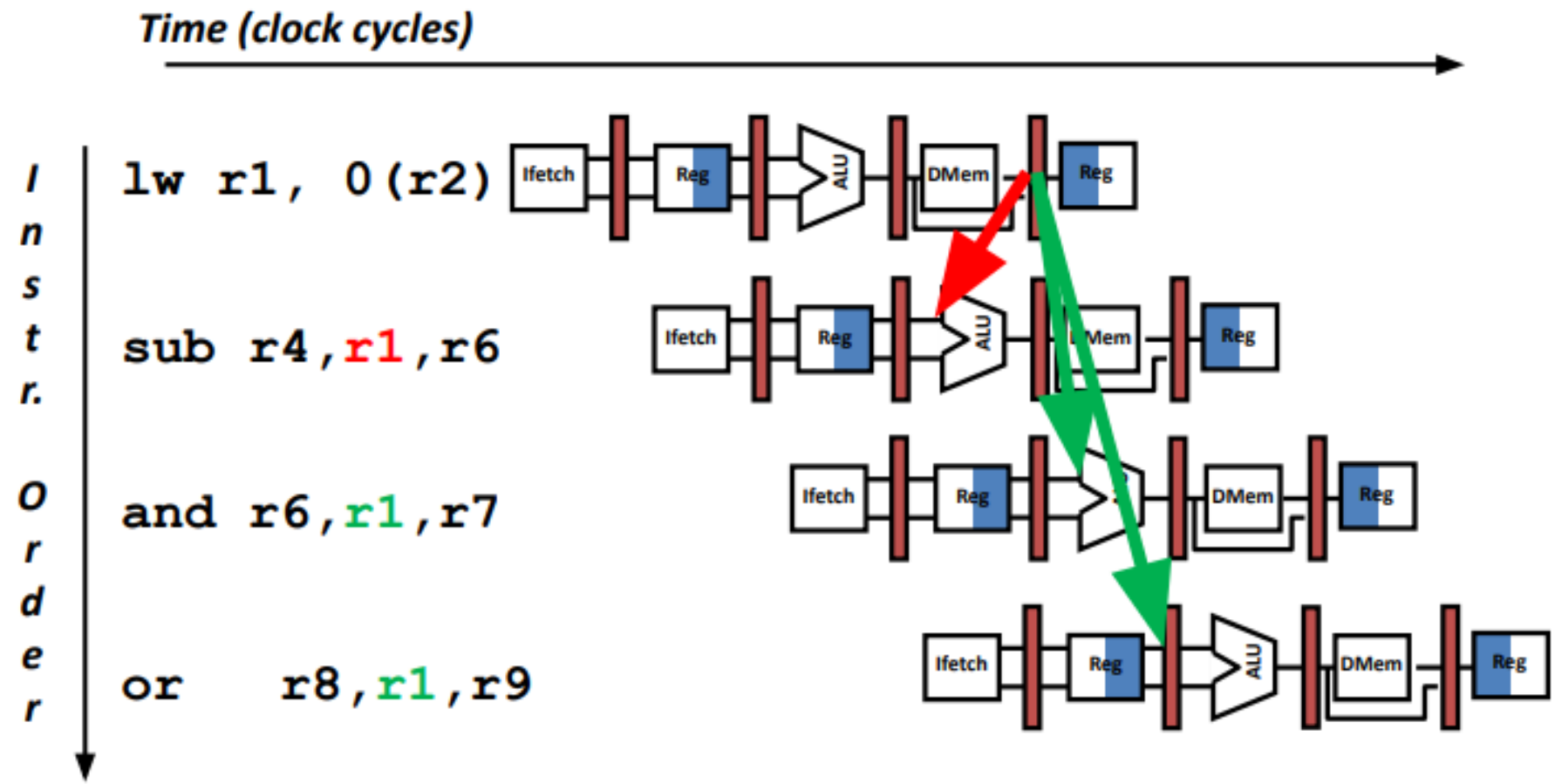
Пример механизма bypassing



Аппаратные изменения data bypassing.



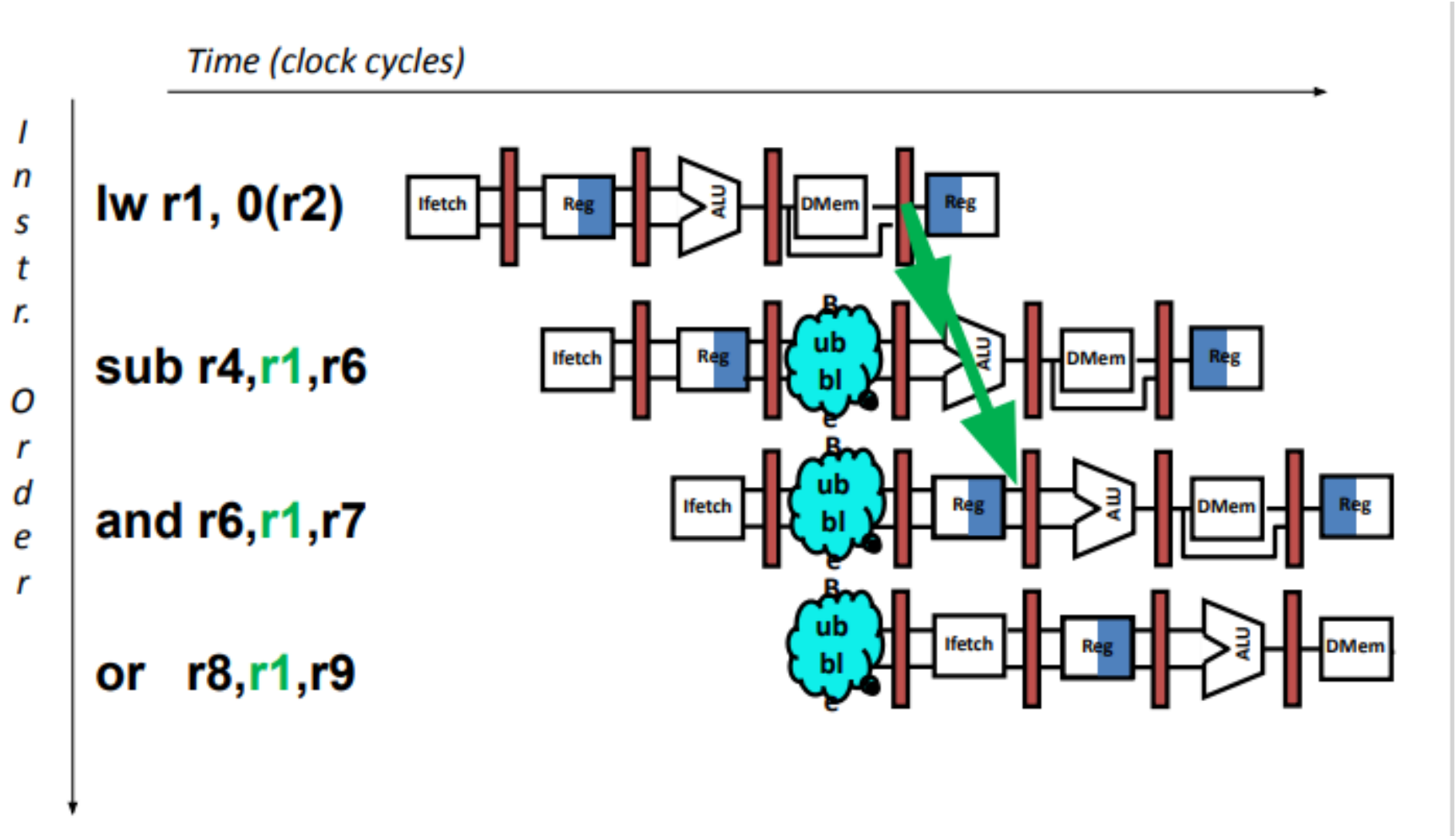
Ограничения data bypassing



Ограничения bypassing

- Данные могут быть переданы на другую стадию, только в том случае, если они реально существуют.
- Необходимым и достаточным условием работы bypass механизма является наличие требуемых данных хотя бы на одной стадии конвейера к тому моменту, когда они действительно потребуются.

Комбинируем простой и bypassing



Внеочередное выполнение команд.