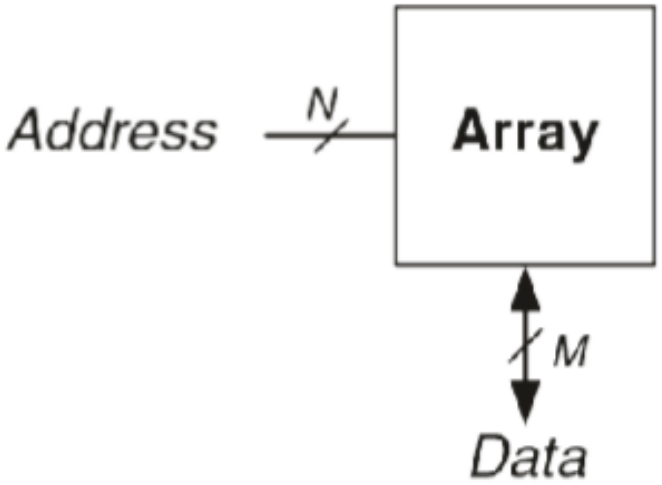
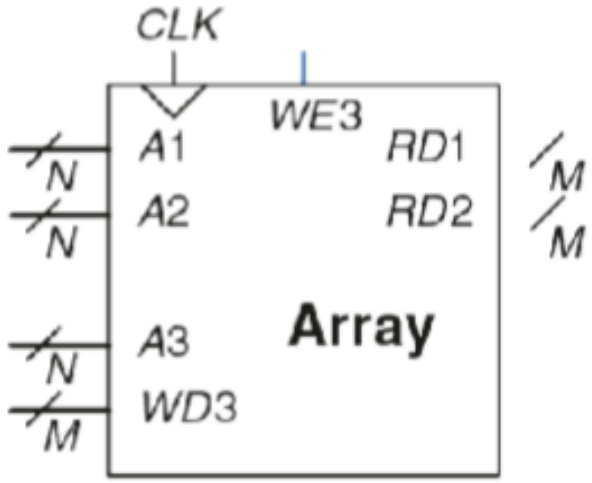
Схемотехника
(углубленный курс)

Лекция №4

Матрицы памяти





ПЗУ

```
module imem(
    input logic [5:0] a,
    output logic [31:0] rd
    );

    logic [31:0] RAM[63:0];

    initial
        $readmemh("memfile.dat", RAM);

    assign rd = RAM[a];

endmodule
```

ОЗУ

```
module dmem(
    input logic clk, we,
    input logic [31:0] a, wd,
    output logic [31:0] rd
    );

    logic [31:0] RAM[63:0];

    assign rd = RAM[a[31:2]];

    always_ff @(posedge clk)
        if (we) RAM[a[31:2]] <= wd;

endmodule
```

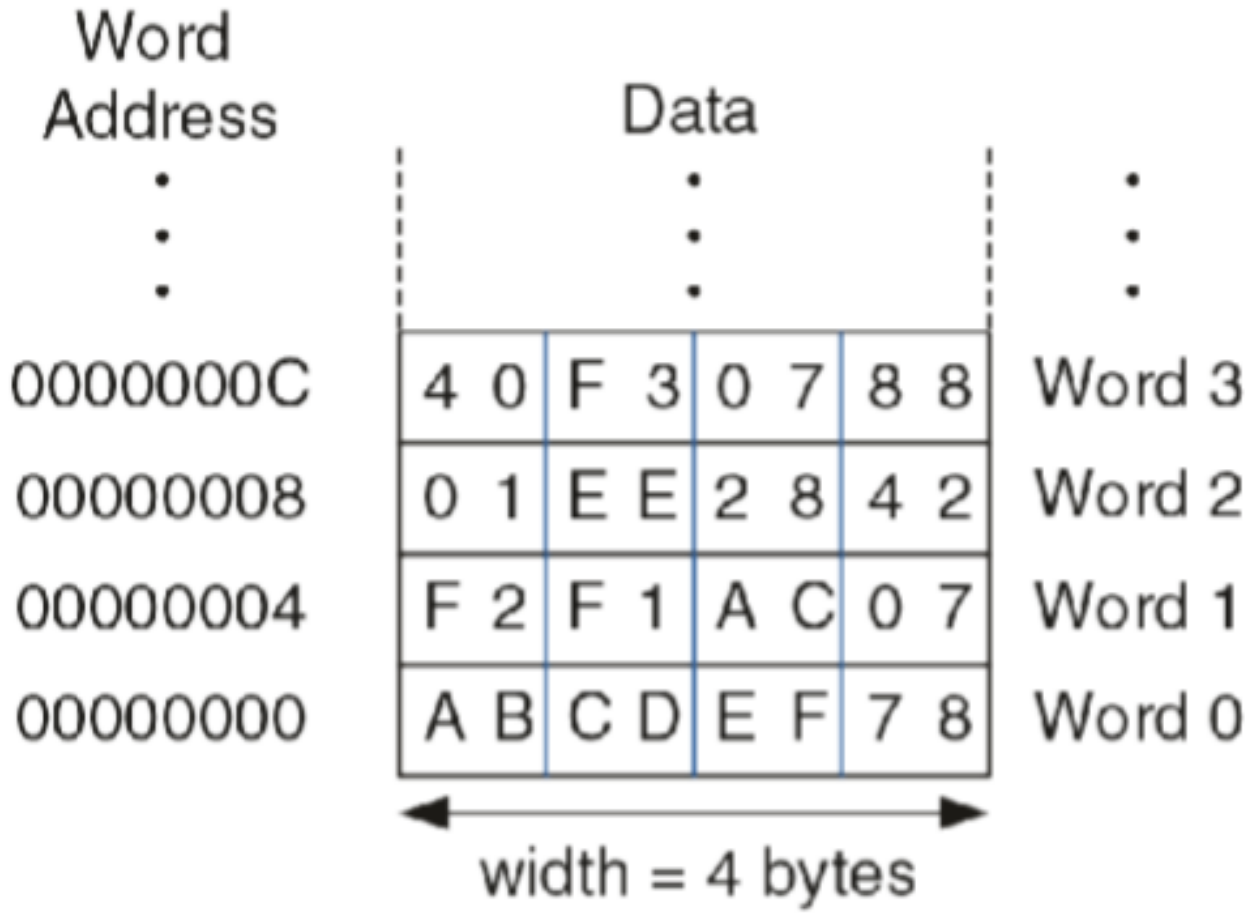
Файл регистров

```
module regfile(
    input logic clk,
    input logic we3,
    input logic [4:0] ra1, ra2, wa3,
    input logic [31:0] wd3,
    output logic [31:0] rd1, rd2
    );

    logic [31:0] rf[31:0];

    always_ff @(posedge clk)
       if (we3) rf[wa3] <= wd3;

    assign rd1 = (ra1 != 0) ? rf[ra1] : 0;
    assign rd2 = (ra2 != 0) ? rf[ra2] : 0;

endmodule
```

Регистры архитектуры MIPS

Название	Номер	Назначение
$0	0	Константный нуль
$at	1	Временный регистр для нужд ассемблера
$v0–$v1	2–3	Возвращаемые функциями значения
$a0–$a3	4–7	Аргументы функций
$t0–$t7	8–15	Временные переменные
$s0–$s7	16–23	Сохраняемые переменные
$t8–$t9	24–25	Временные переменные
$k0–$k1	26–27	Временные переменные операционной системы
$gp	28	Глобальный указатель (англ.: global pointer)
$sp	29	Указатель стека (англ.: stack pointer)
$fp	30	Указатель кадра стека (англ.: frame pointer)
$ra	31	Регистр адреса возврата из функции

Память с побайтовой адресацией



Доступ к памяти

```
lw  $s0, 0($0)          # read data word 0 (0xABCDEF78) into $s0
lw  $s1, 8($0)          # read data word 2 (0x01EE2842) into $s1
lw  $s2, 0xC($0)        # read data word 3 (0x40F30788) into $s2
sw  $s3, 4($0)          # write $s3 to data word 1
sw  $s4, 0x20($0)       # write $s4 to data word 8
sw  $s5, 400($0)        # write $s5 to data word 100
```

Непосредственные операнды

Код на языке высокого уровня

```
a = a + 4;
b = a - 12;
```

Код на языке ассемблера MIPS

```
# $s0 = a, $s1 = b
  addi  $s0, $s0, 4          # a = a + 4
  addi  $s1, $s0, -12        # b = a - 12
```

Формат команды типа R

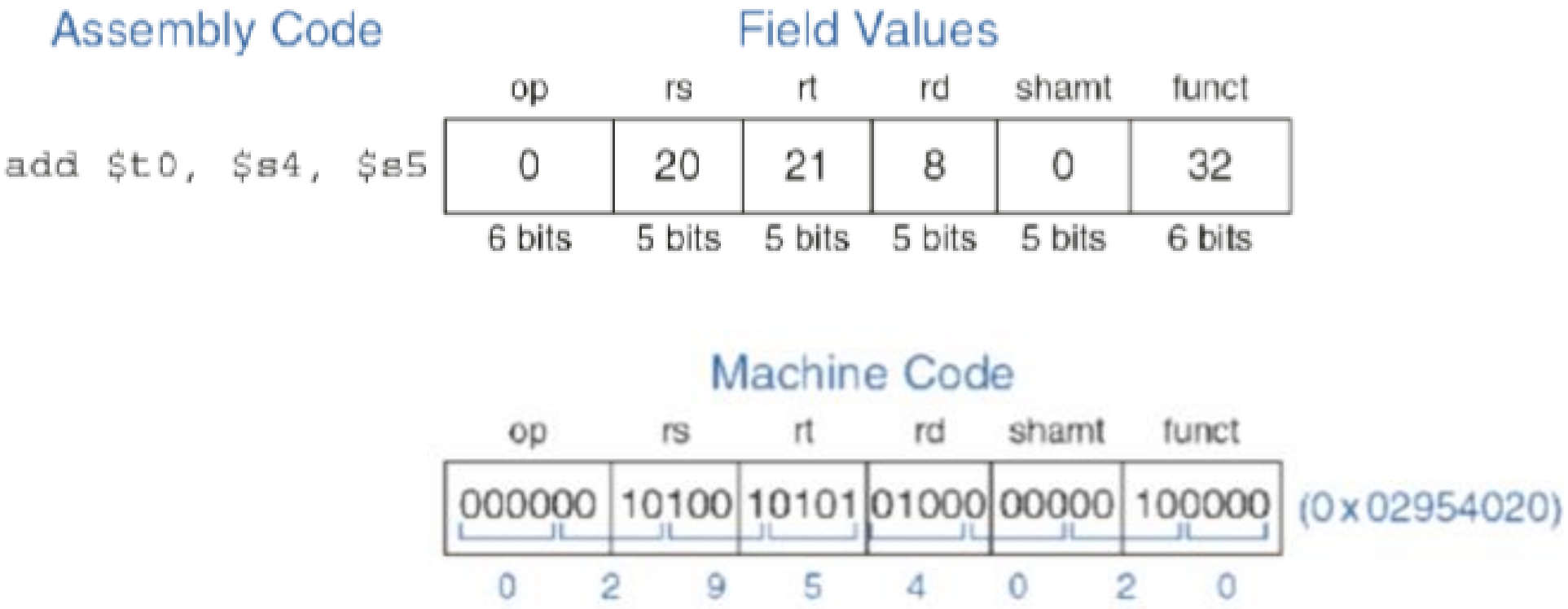
Примеры команд типа R

100000 (32)	`add rd, rs, rt`	сложение со знаком, арифметическое переполнение вызвает исключение (add)	`[rd] = [rs] + [rt]`
100010 (34)	`sub rd, rs, rt`	Вычитание со знаком, арифметическое переполнение вызвает исключение (subtract)	`[rd] = [rs] - [rt]`
100100 (36)	`and rd, rs, rt`	побитовое логическое «И» (and)	`[rd] = [rs] & [rt]`
100101 (37)	`or rd, rs, rt`	побитовое логическое «ИЛИ» (or)	`[rd] = [rs] | [rt]`

Битовый сдвиг

Funct	Имя	Описание	Операция
000000 (0)	`sll rd, rt, shamt`	логический сдвиг влево (shift left logical)	`[rd] = [rt] << shamt`
000010 (2)	`srl rd, rt, shamt`	логический сдвиг вправо (shift right logical)	`[rd] = [rt] >> shamt`
000011 (3)	`sra rd, rt, shamt`	арифметический сдвиг вправо (shift right arithmetic)	`[rd] = [rt] >>> shamt`
000100 (4)	`sllv rd, rt, rs`	логический переменный сдвиг влево (shift left logical variable)	$[rd] = [rt] << [rs]_{4:0}$
000110 (6)	`srlv rd, rt, rs`	логический переменный сдвиг вправо (shift right logical variable)	$[rd] = [rt] >> [rs]_{4:0}$
000111 (7)	`srav rd, rt, rs`	арифметический переменный сдвиг вправо (shift right arithmetic variable)	$[rd] = [rt] >>> [rs]_{4:0}$

Пример трансляции команды типа R



Формат команды типа I

Примеры опкодов команд типа I

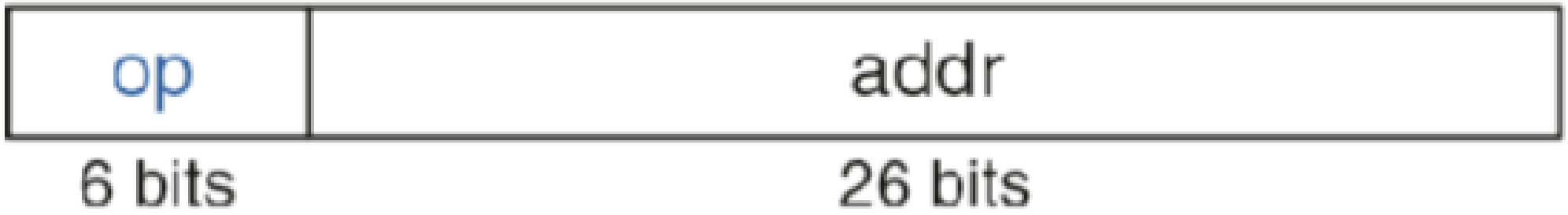
Opcode	Имя	Описание	Операция
001000 (8)	`addi rt, rs, imm`	сложение с непосредственным операндом со знаком, арифметическое переполнение вызывает исключение (add immediate)	`[rt] = [rs] + SignImm`
001101 (13)	`ori rt, rs, imm`	побитовое логическое «ИЛИ» с непосредственным операндом, расширенным влево нулями (or immediate)	`[rt] = [rs] |ZeroImm`
100011 (35)	`lw rt, imm(rs)`	загрузка слова (load word)	`[rt] = [Address]`
101011 (43)	`sw rt, imm(rs)`	сохранение слова (store word)	`[Address] = [rt]`

```
addi $s0, $s1, 5
addi $t0, $s3, -12
lw   $t2, 32($0)
sw   $s1,  4($t1)
```

op	rs	rt	imm
8	17	16	5
8	19	8	–12
35	0	10	32
43	9	17	4
6 bits	5 bits	5 bits	16 bits

op	rs	rt	imm	
001000	10001	10000	0000 0000 0000 0101	(0x22300005)
001000	10011	01000	1111 1111 1111 0100	(0x2268FFF4)
100011	00000	01010	0000 0000 0010 0000	(0x8C0A0020)
101011	01001	10001	0000 0000 0000 0100	(0xAD310004)
6 bits	5 bits	5 bits	16 bits	

Формат команды типа J



Примеры опкодов команд переходов

000001 (1) (rt = 0/1)	`bltz rs, label /` `bgez rs, label`	ветвление, если меньше нуля (branch less than zero) / ветвление, если больше или равно нулю (branch greater than or equal to zero)	`if ([rs] < 0) PC = BTA/` `if ([rs] ≥ 0) PC = BTA`
000010 (2)	`j label`	безусловный переход (jump)	`PC = JTA`
000011 (3)	`jal label`	безусловный переход с возвратом (jump and link)	`$ra = PC + 4, PC = JTA`
000100 (4)	`beq rs, rt, label`	ветвление, если равно (branch if equal)	`if ([rs] == [rt])` `PC = BTA`

Режимы адресации

- Регистровая адресация (инструкции типа R)
- Непосредственная адресация (16-битные константы некоторых инструкций типа I)
- Базовая адресация (база из регистра плюс 16-битное смещение)
- Адресация относительно счетчика команд (BTA)
- Псевдопрямая адресация (JTA)

Вычисление адресов переходов

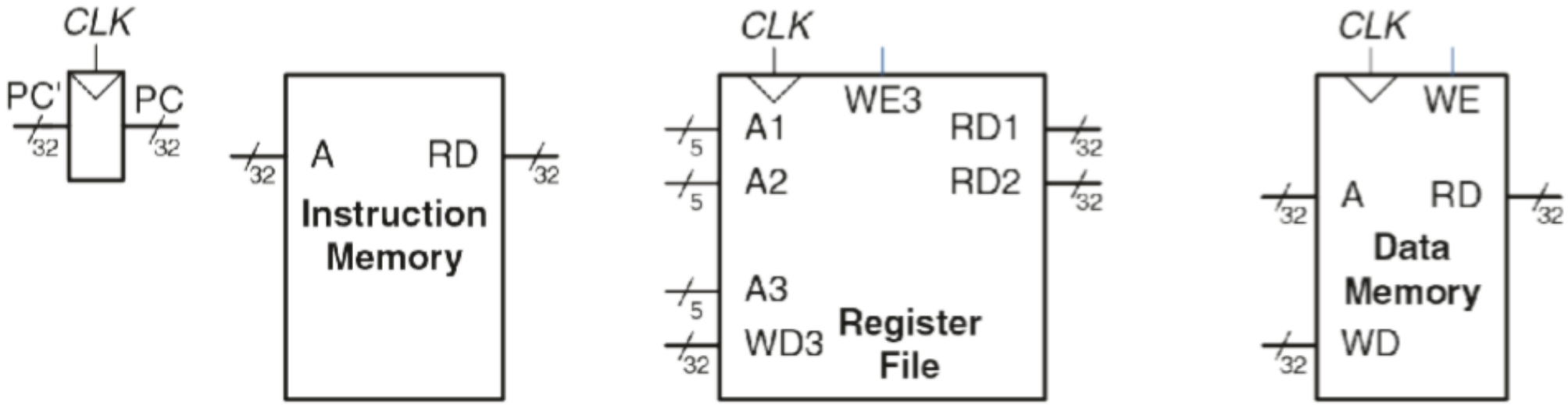
BTA: **целевой адрес ветвления**

```
= PC + 4 + (SignImm << 2)
```

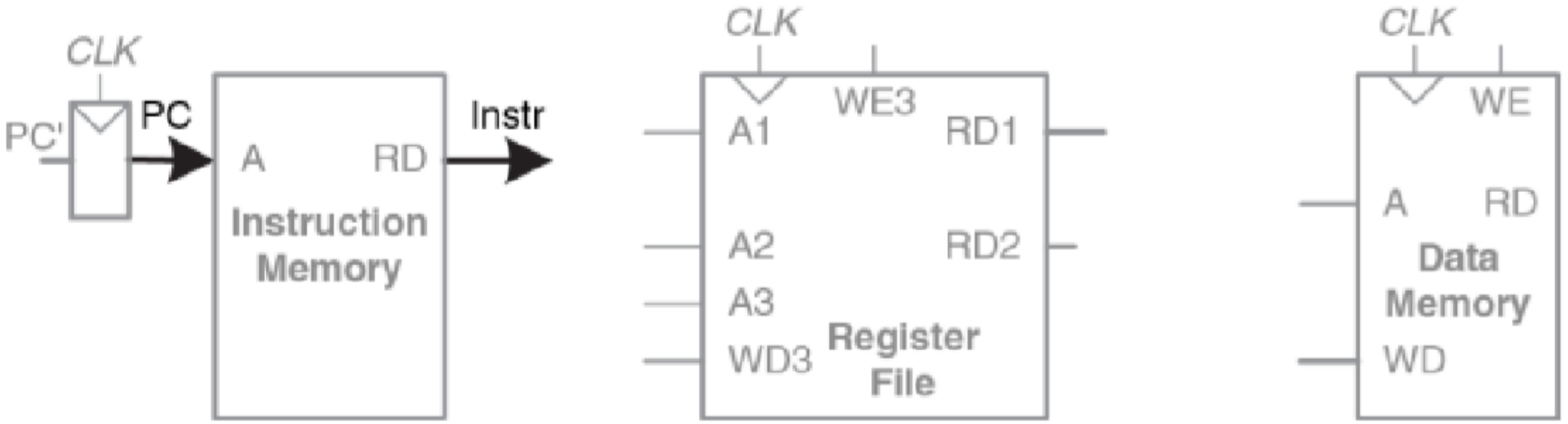
JTA: **адрес перехода**

```
= {(PC + 4)[31:28], addr, 2'b0}
```

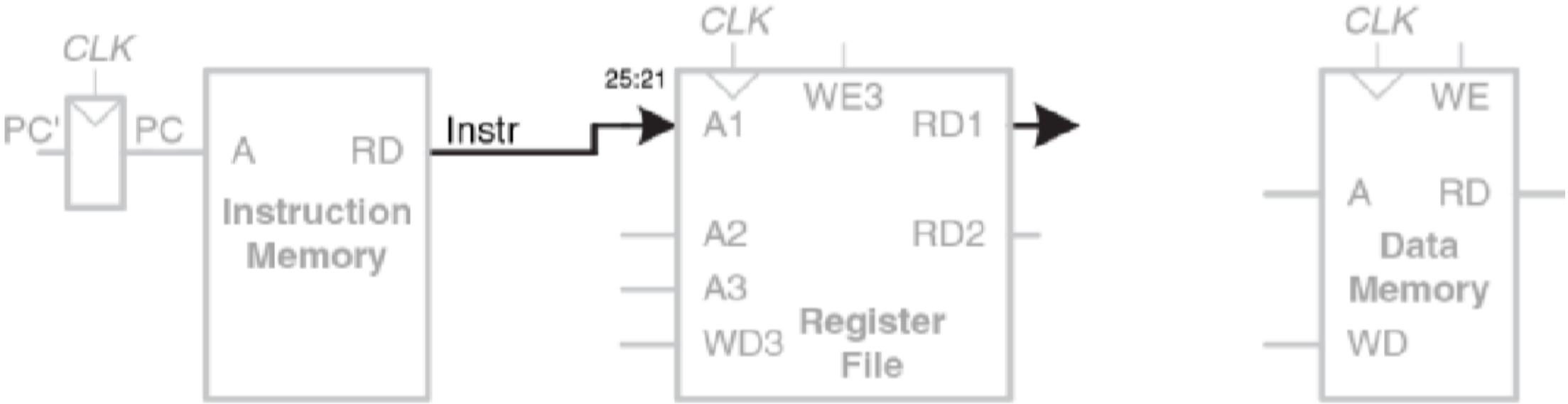
Элементы, хранящие состояние процессора



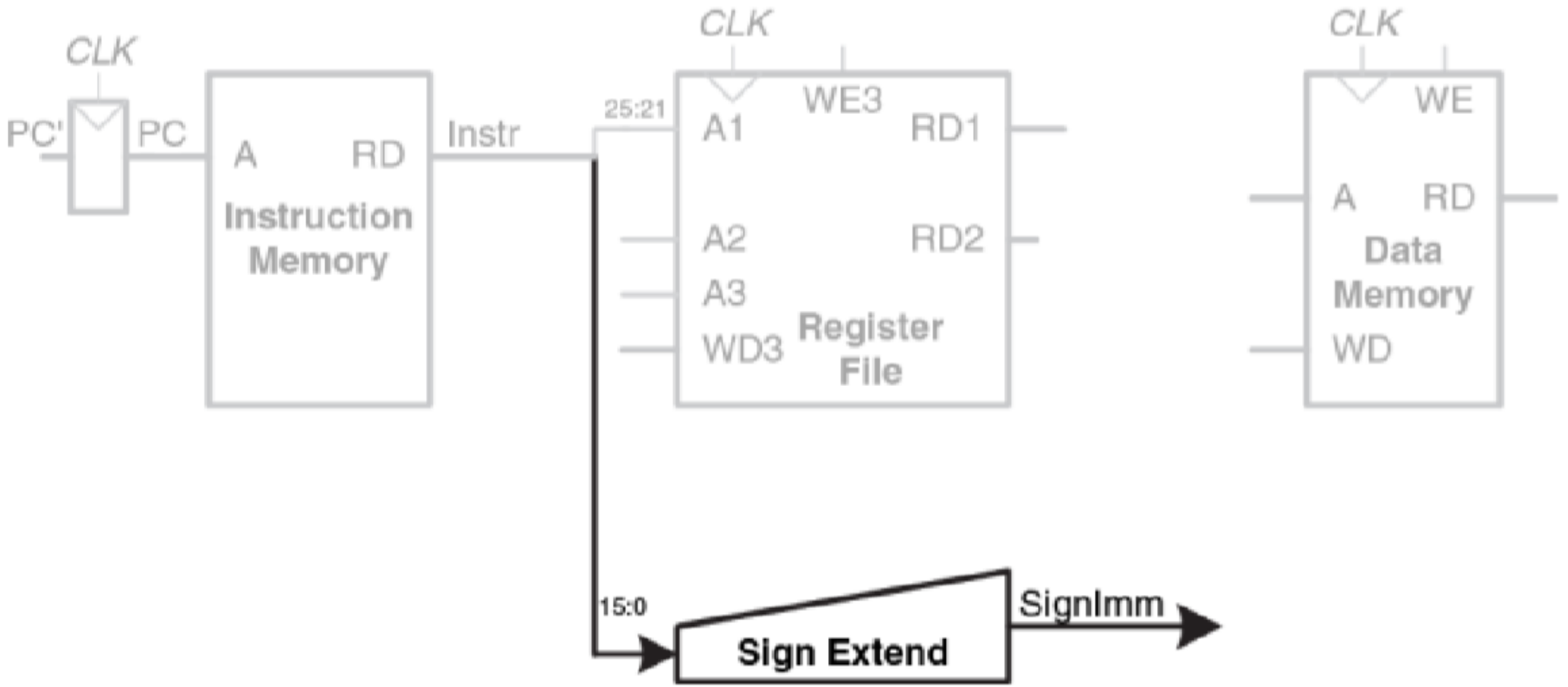
Выборка команды из памяти



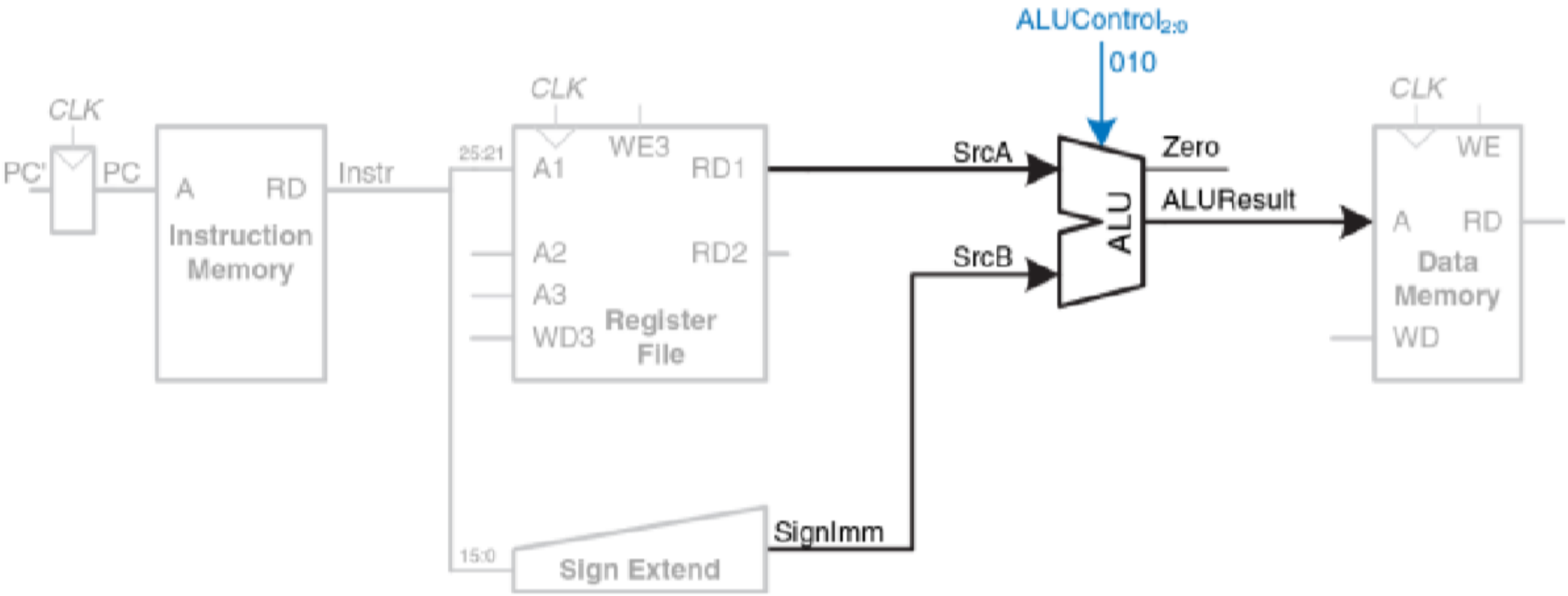
Чтение операнда из регистрового файла



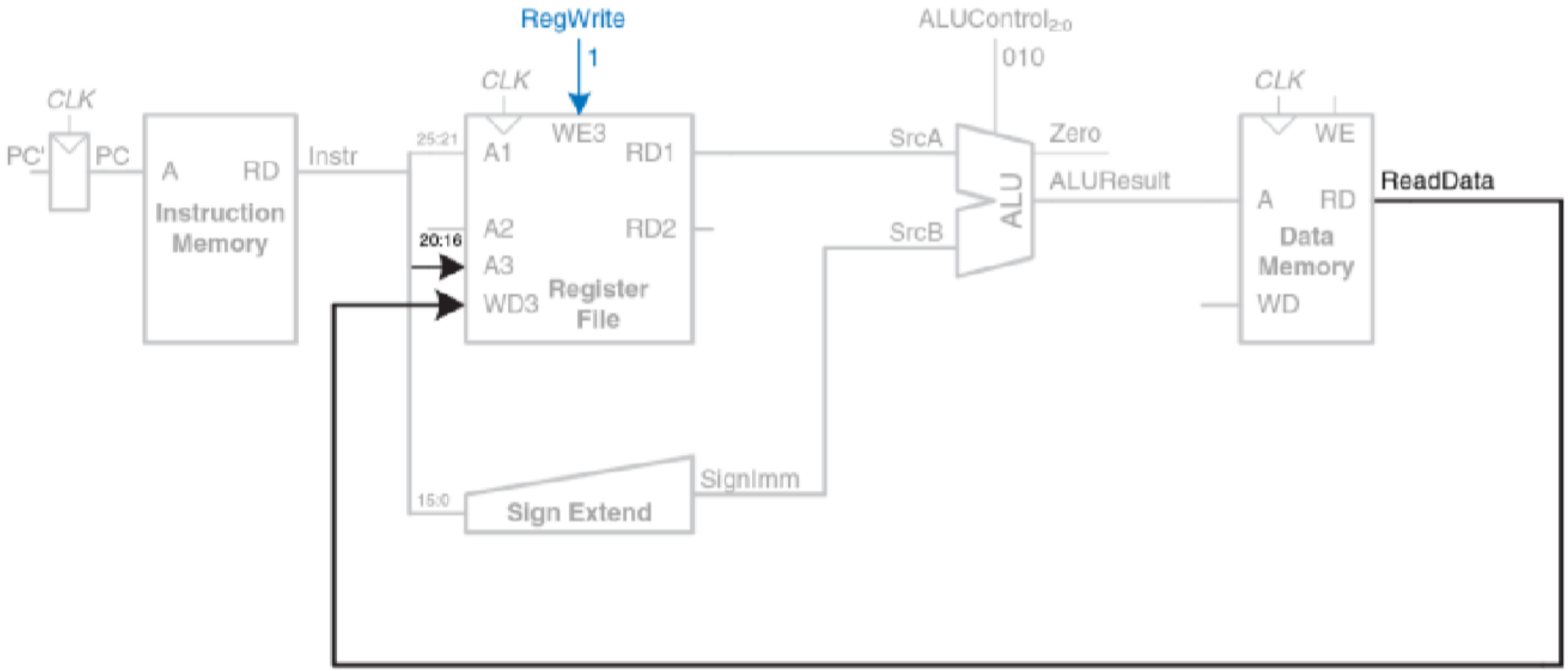
Знаковое расширение непосредственного операнда



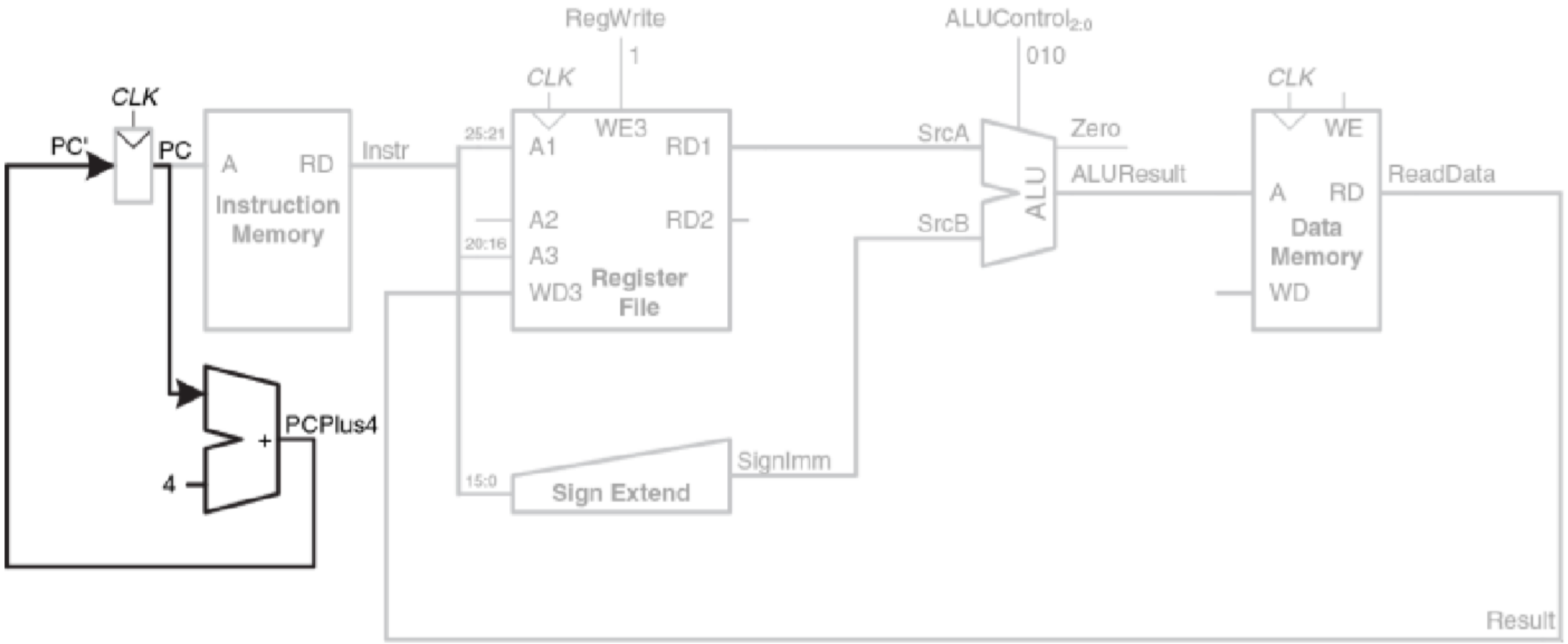
Вычисление адреса данных в памяти



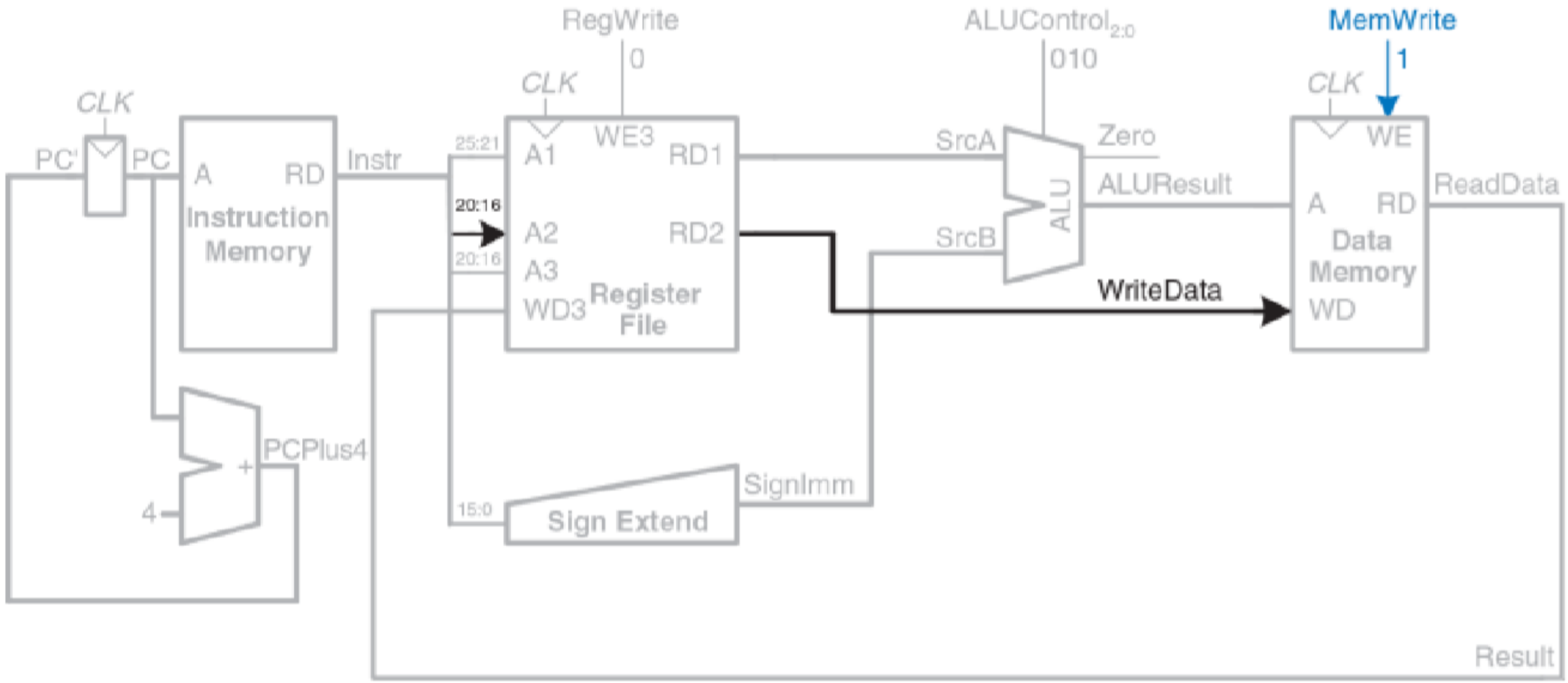
Запись в регистровый файл



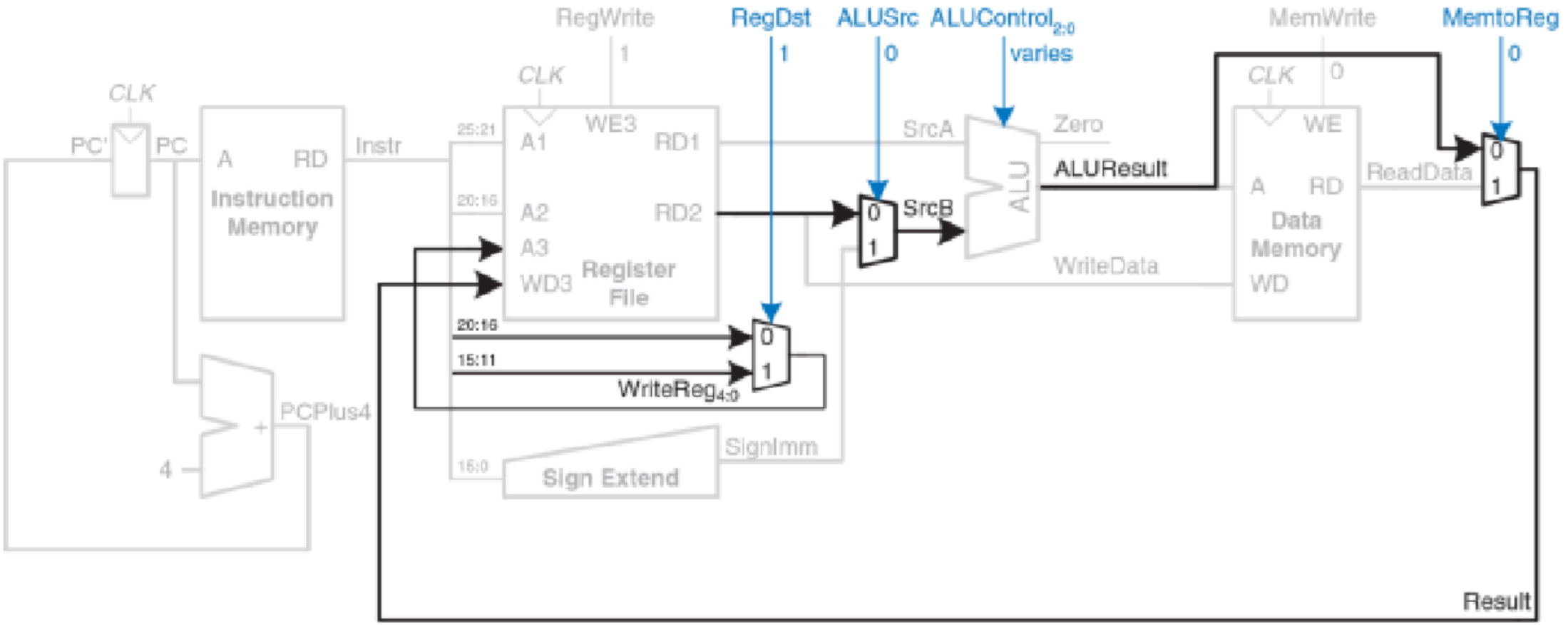
Определение адреса следующей команды



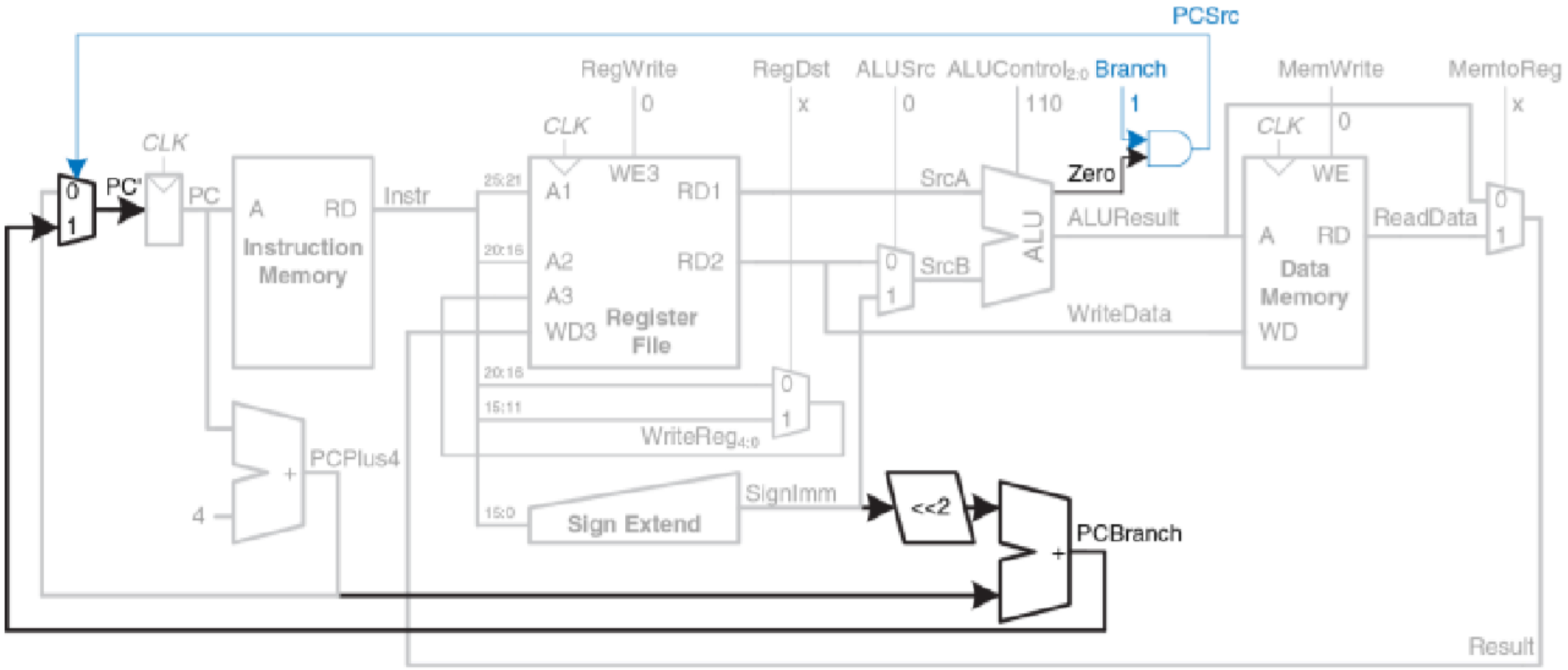
Запись командой **SW** данных в память



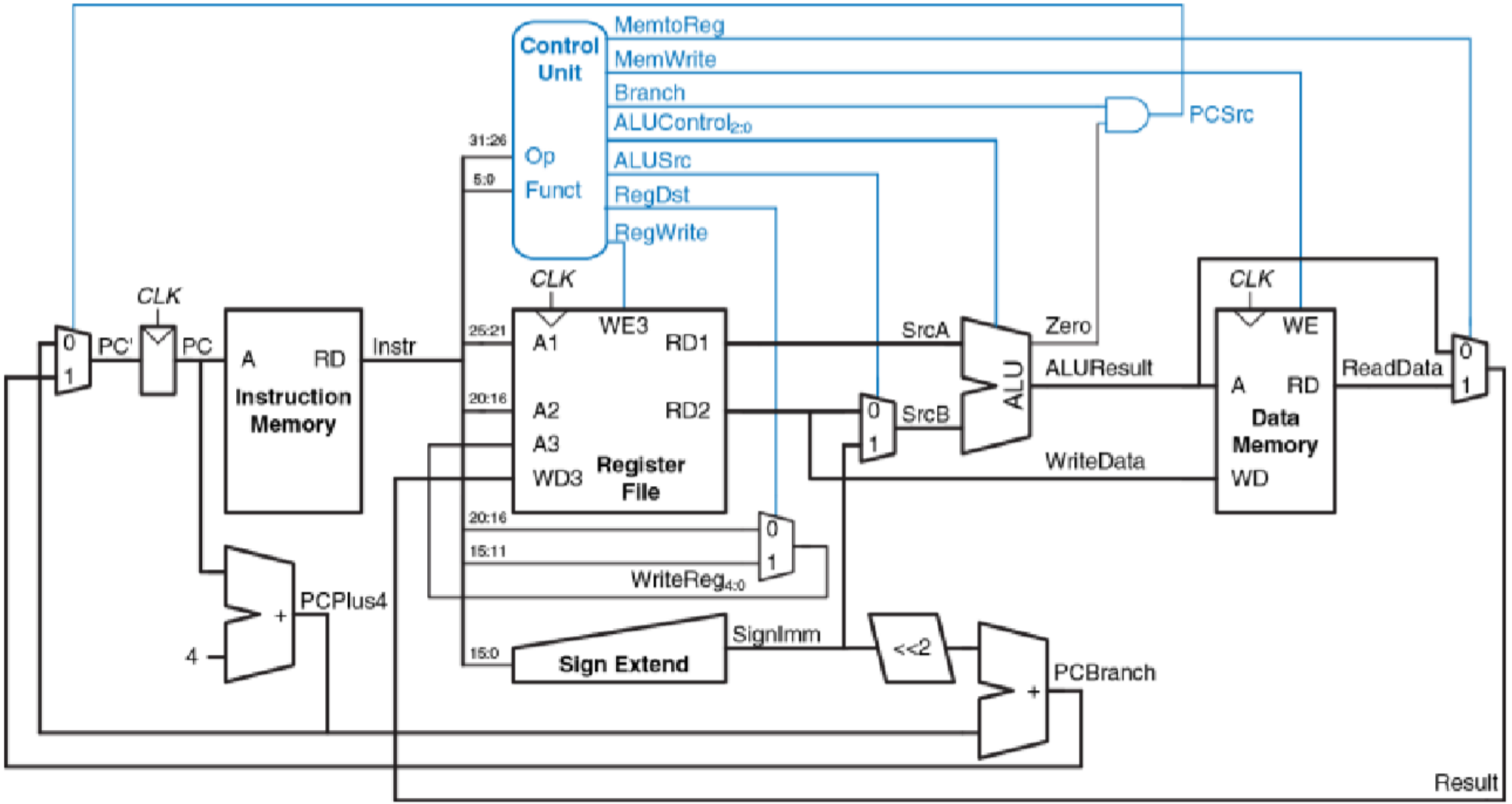
Поддержка команд типа **R**



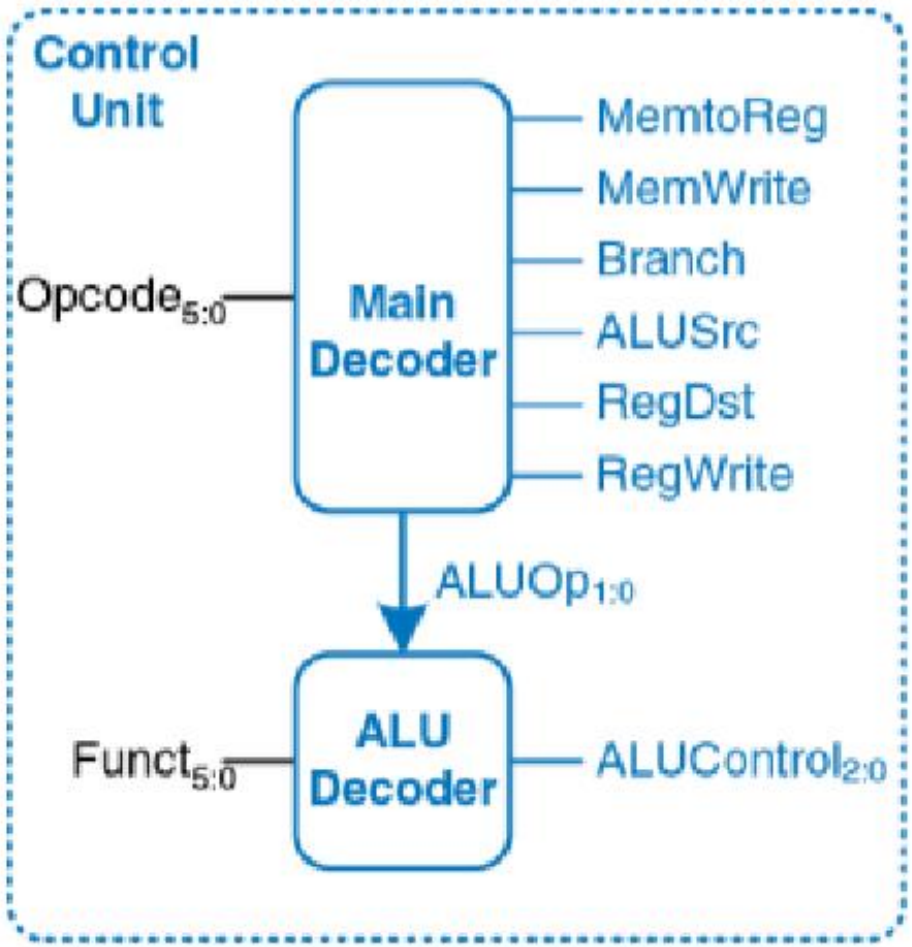
Поддержка команды **beq**



Однотактный процессор



Устройство управления



ALUOp	Функция
00	Сложение
01	Вычитание
10	Определяется полем `funct`
11	Не используется

ALUOp	funct	ALUControl
00	X	010 (сложение)
X1	X	110 (вычитание)
1X	100000 (`add`)	010 (сложение)
1X	100010 (`sub`)	110 (вычитание)
1X	100100 (`and`)	000 (логическое «И»)
1X	100101 (`or`)	001 (логическое «ИЛИ»)
1X	101010 (`slt`)	111 (установить, если меньше)

Команда	Opcode	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemtoReg	ALUOp
Команды типа R	000000	1	1	0	0	0	0	10
`lw`	100011	1	0	1	0	0	1	00
`sw`	101011	0	X	1	0	1	X	00
`beq`	000100	0	X	0	1	0	X	01