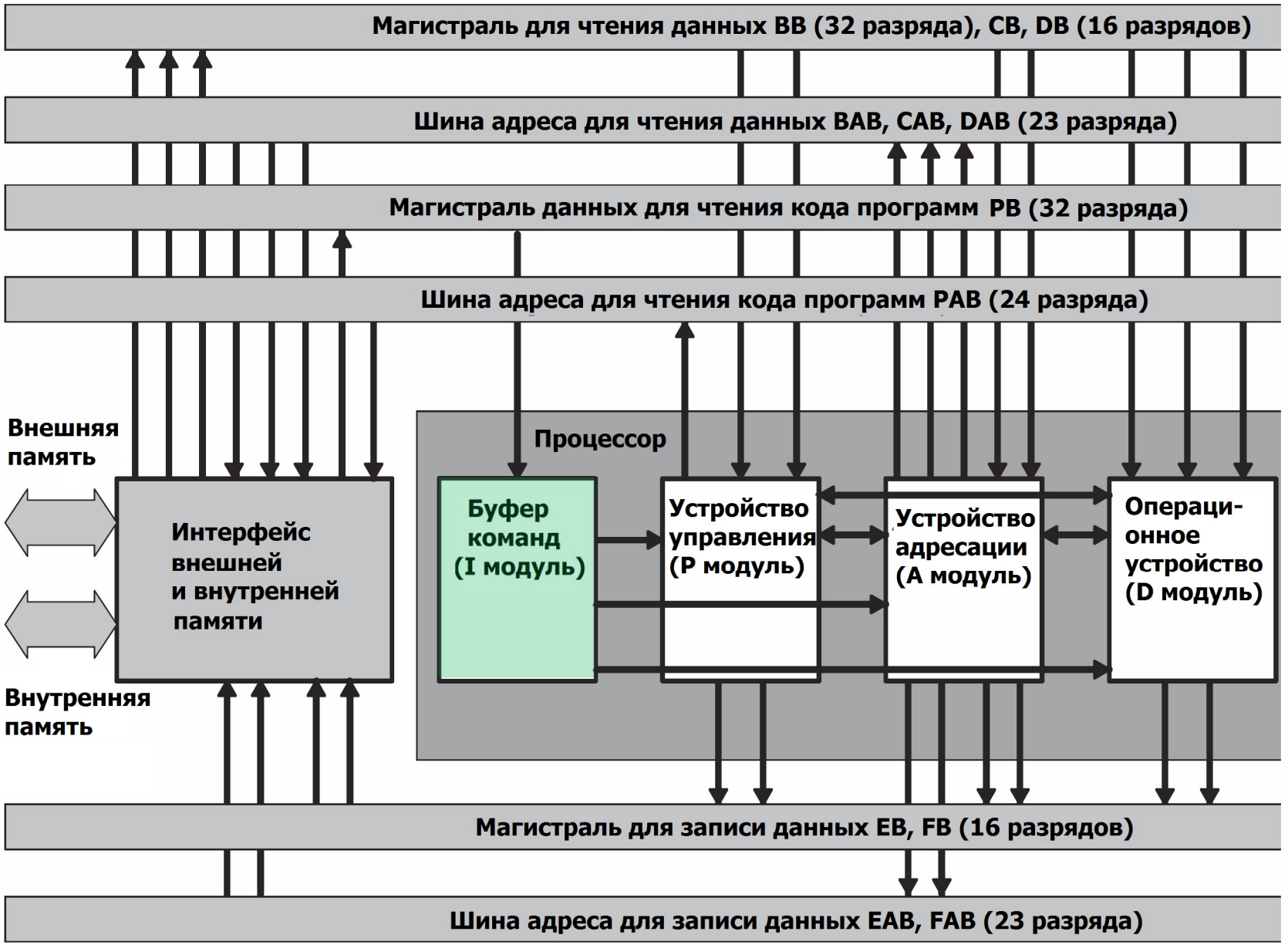
Микропроцессорные устройства обработки сигналов

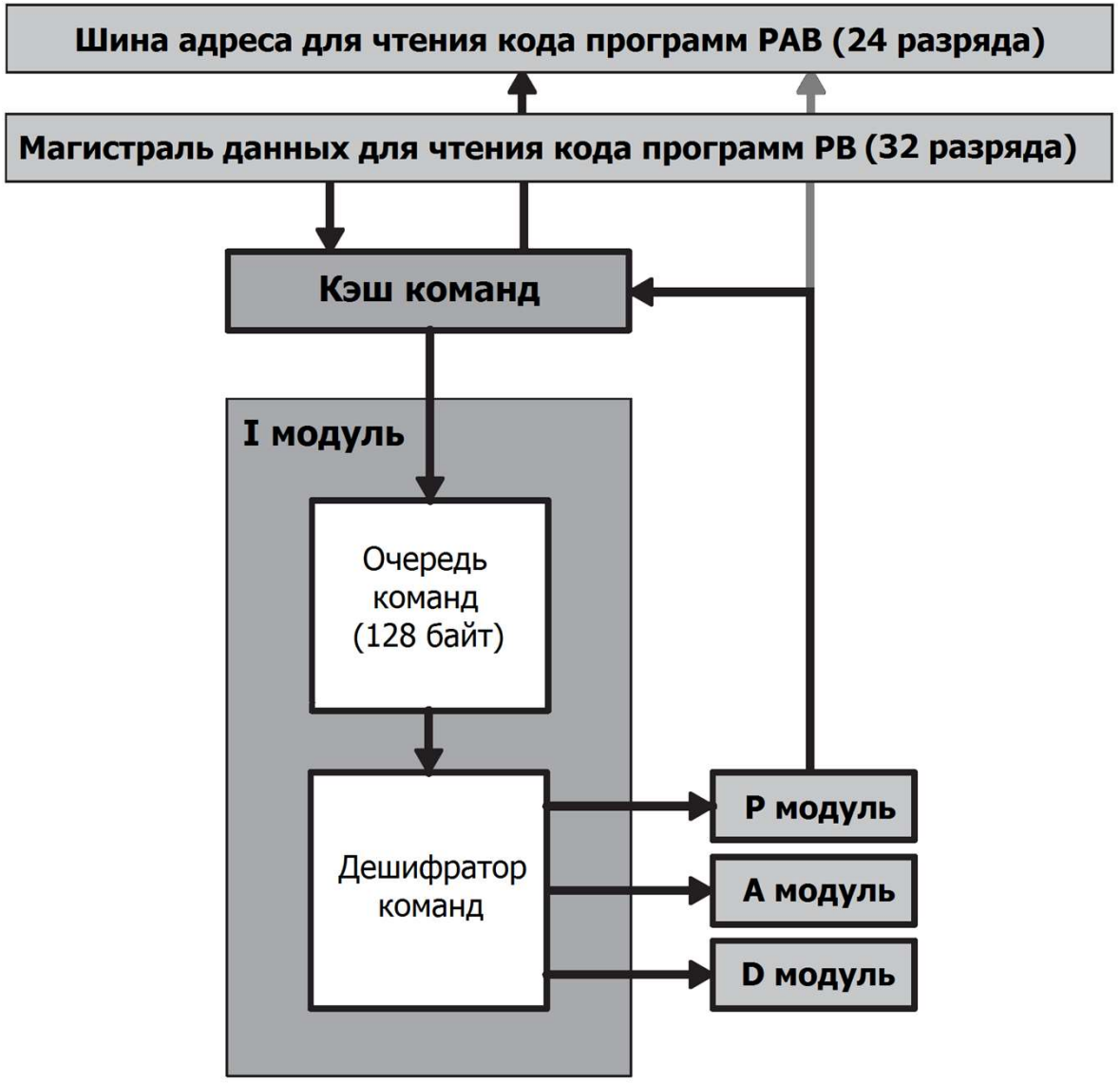
Лекция L09
«Конвейеризация и распараллеливание»

http://vykhovanets.ru/course67/

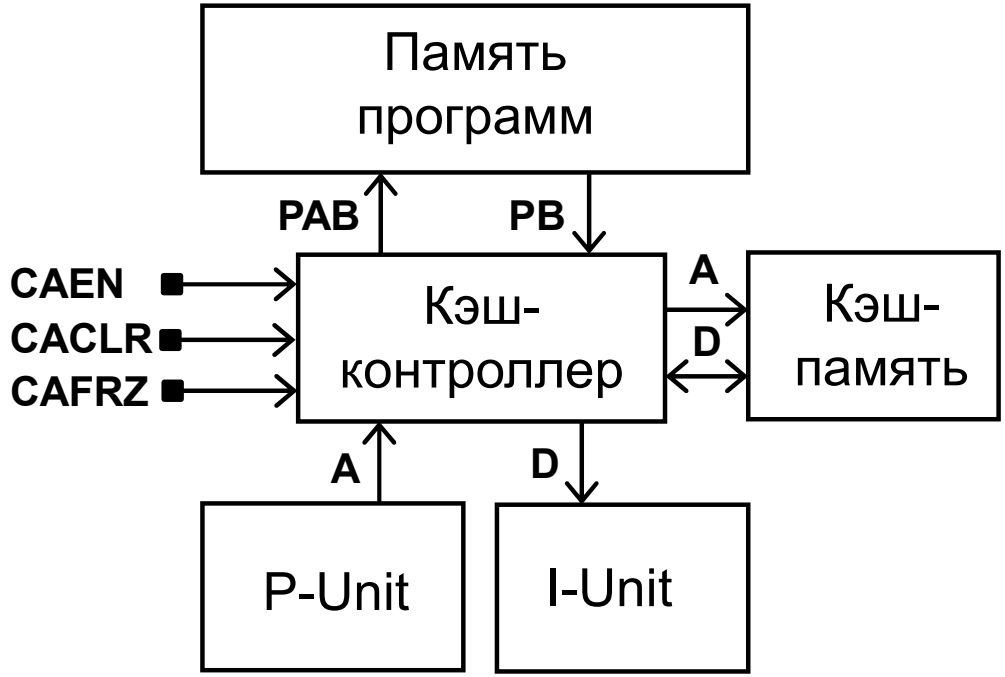
Ядро микропроцессора

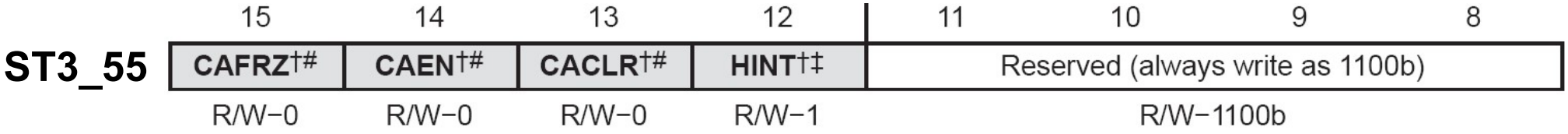


Буфер команд



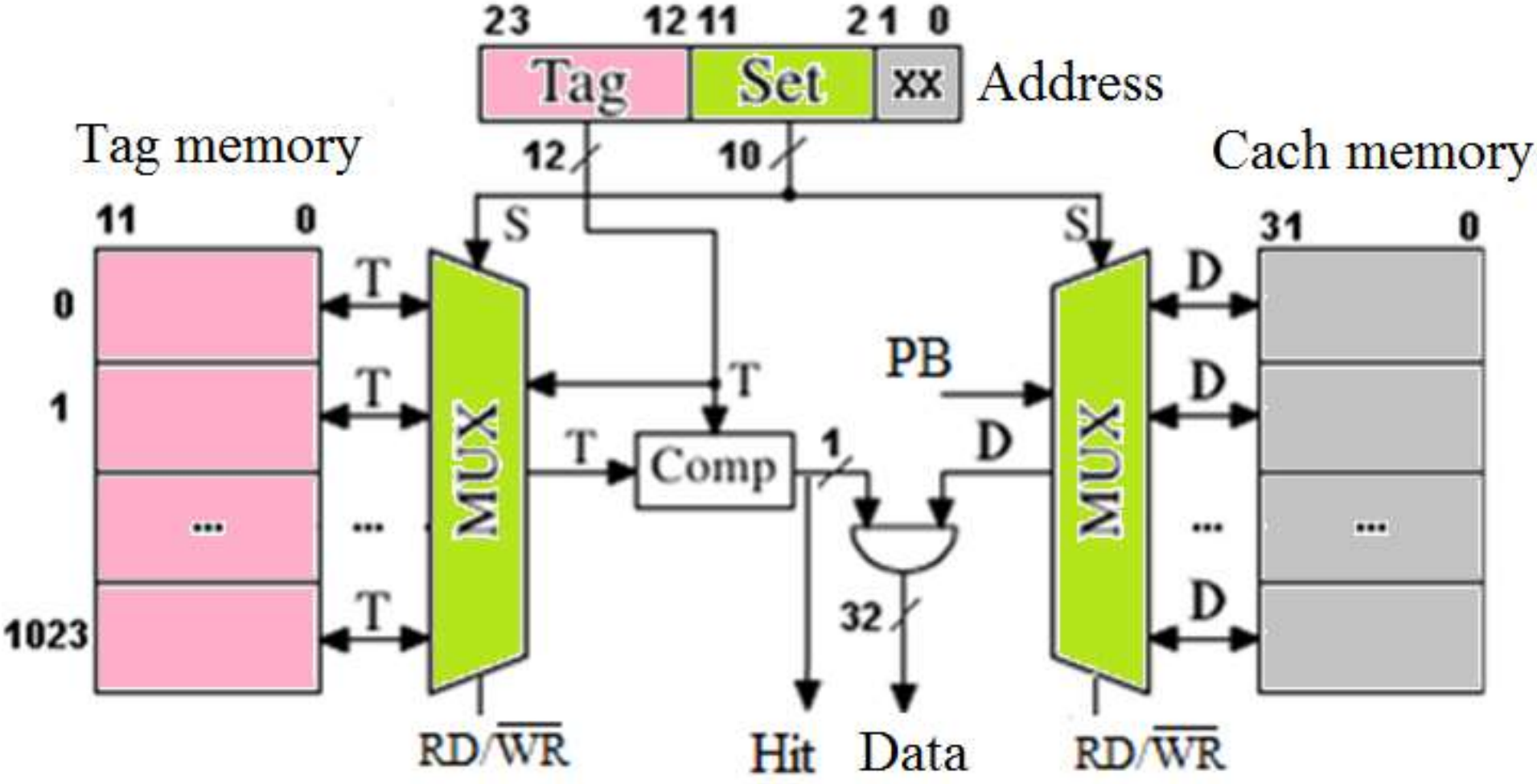
Кэш команд





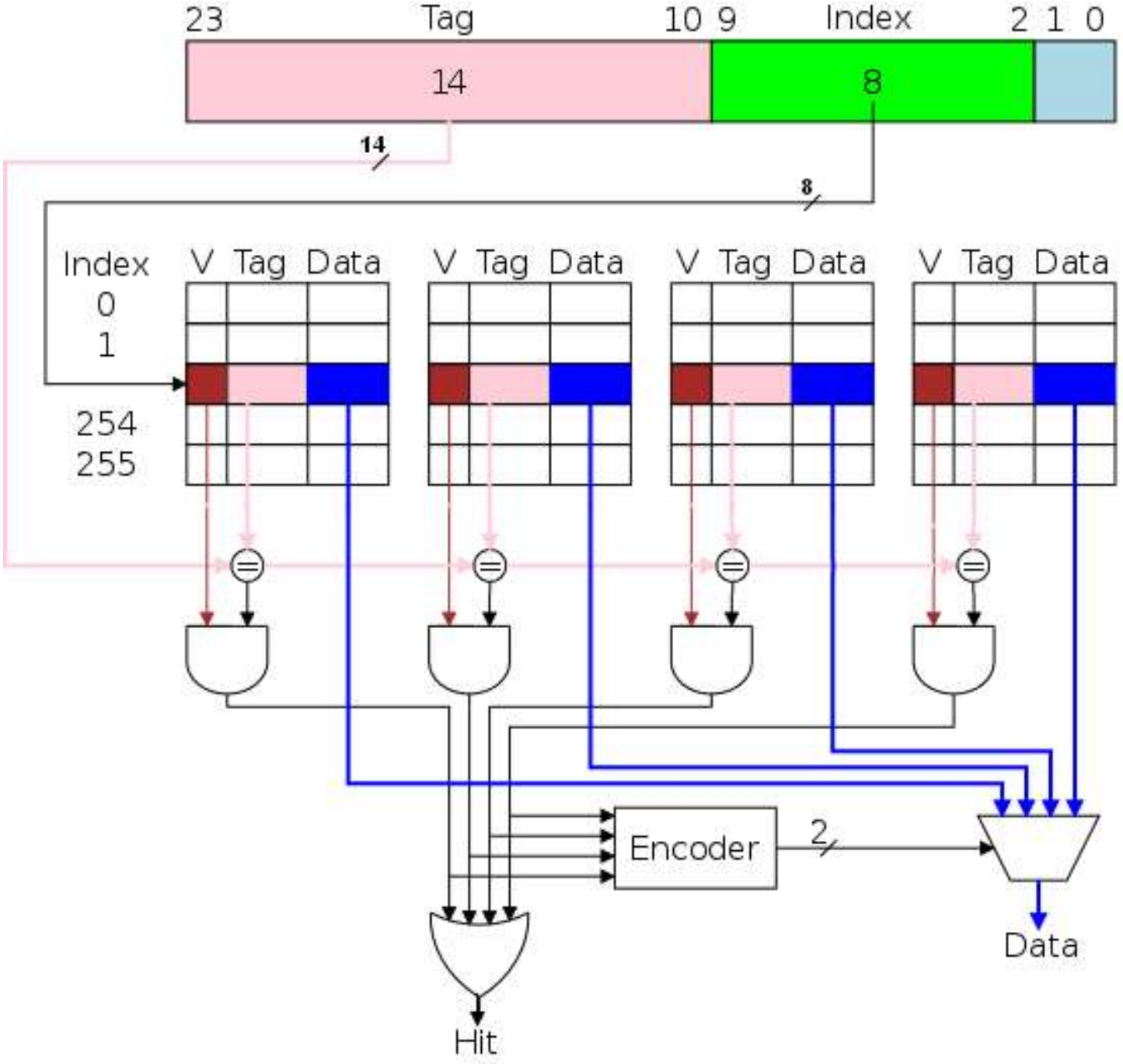
- **Cache** – тайник (кэш)
- **CACLR** – **Ca**sh **Cl**ea**r** (флаг очистки).
- **CAEN** – **Ca**sh **En**able (флаг разрешения).
- **CAFRZ** – **Ca**sh **Fr**ee**z**e (флаг замораживания).

Кэш прямого отображения



- **Tag** – признак
- **Set** – набор данных, место в кэш-памяти
- **RD, WR** – Чтение или запись в кэш-память (направление пересылки данных и тегов)

Ассоциативная кэш-память

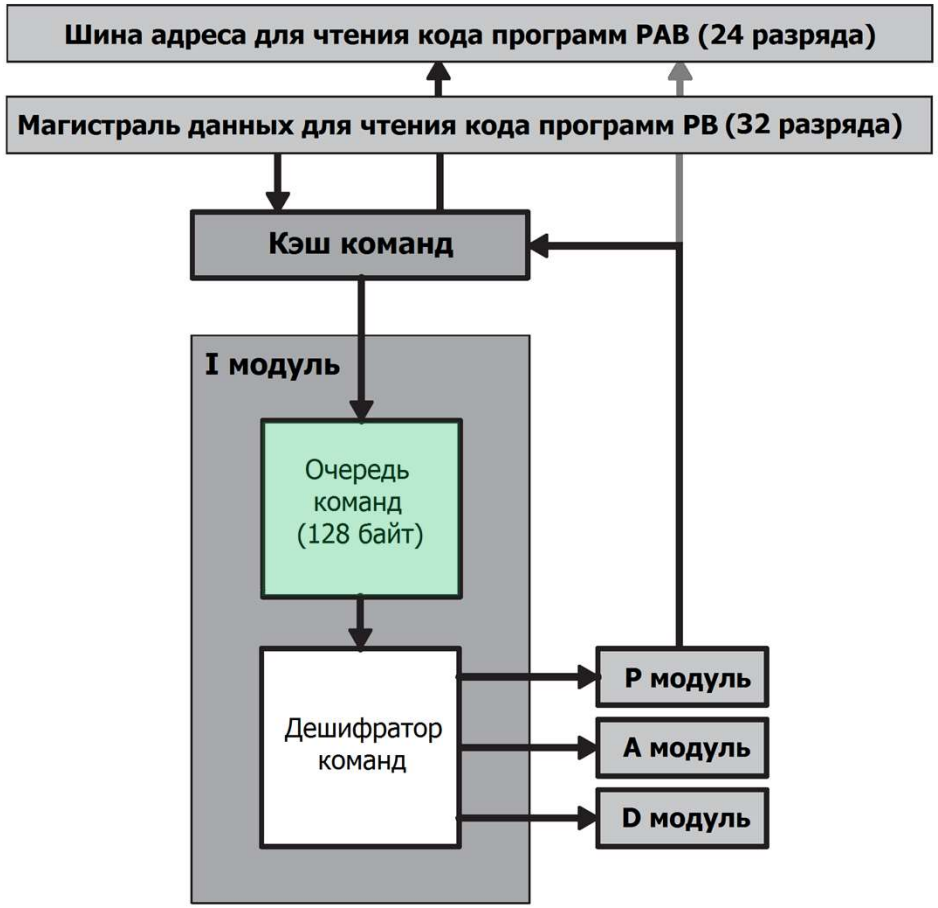


- **C5501, C5502** – ассоциативная кэш-память с двумя наборами

- **C5510** –
а) режим ассоциативной кэш-памяти с двумя наборами

б) режим кэш-памяти с прямым отображением

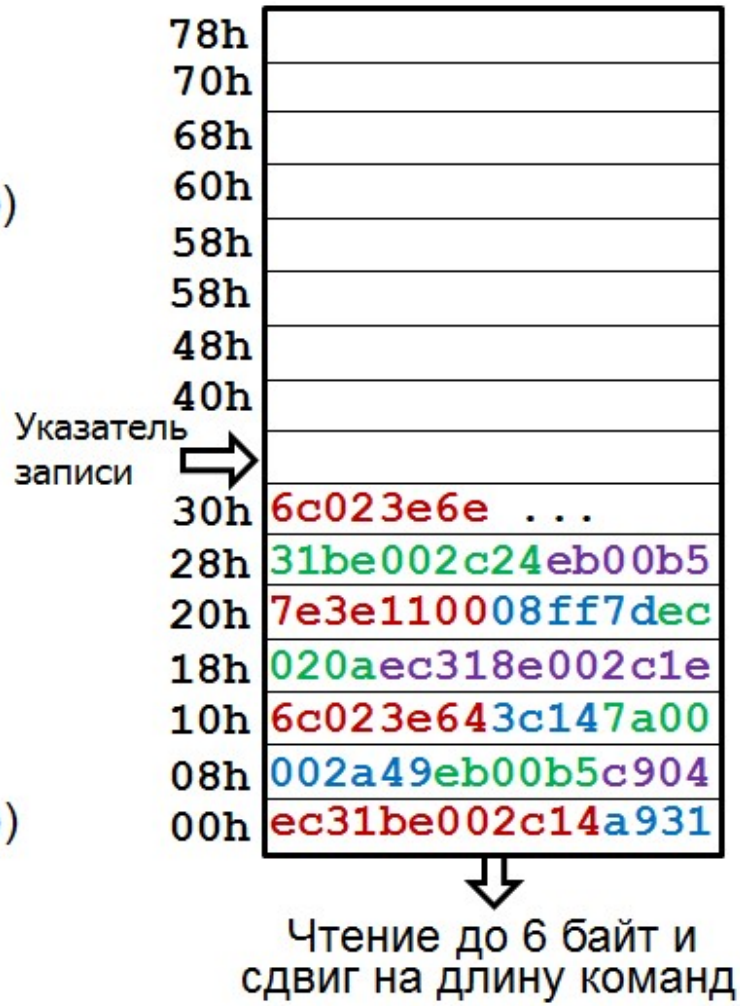
Очередь команд



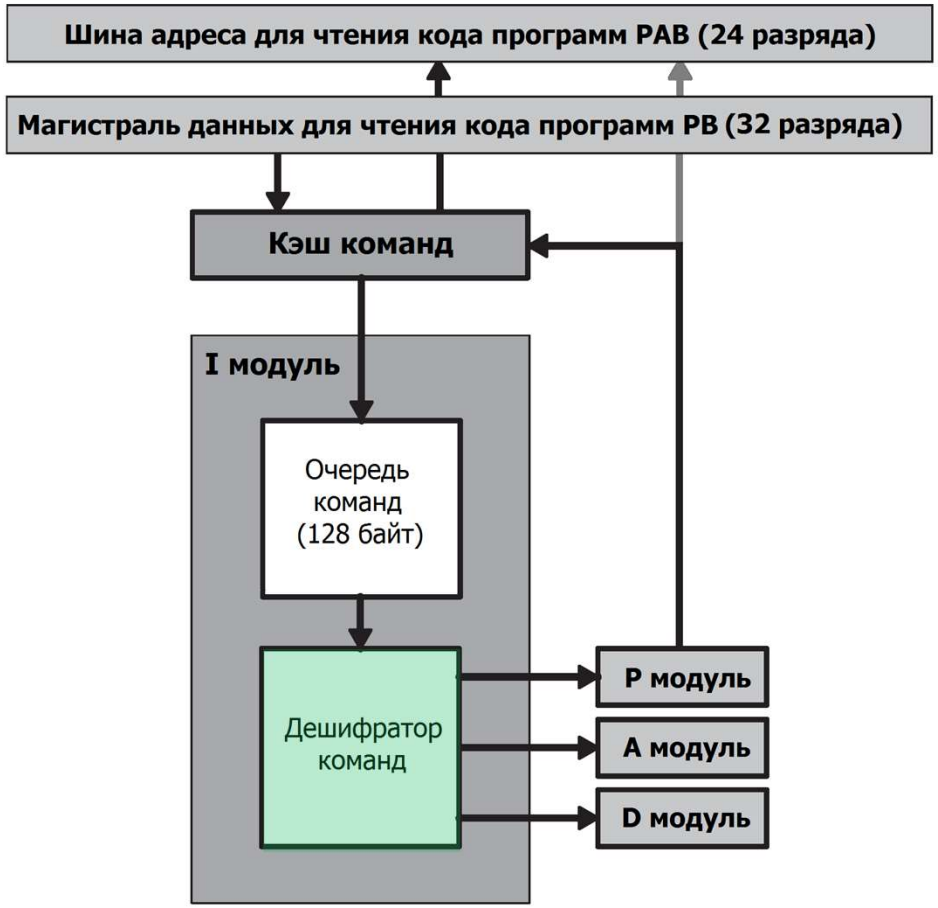
- Загрузка 32-битных слов в конец очереди.
- Сдвиг команд в буфере к началу очереди.
- Передача первых 6 байт в декодер команд.
- Хранение локальных блоков повторения (до 128 байт).
- Очистка буфера при ветвлениях в программе (переходы, вызов и возврат подпрограмм, прерывания)

Заполнение очереди

```
/*          printf("\n%04x  T0", test);                */
02399D:   ec31be002c14  AMAR *(#02c14h),XAR3
0239A3:   a931002a49    MOV *(#02a49h),AR1
0239A8:   eb00b5        MOV XAR3,dbl(*SP(#00h))
0239AB:   c904          MOV AR1,*SP(#02h)
0239AD:   6c023e64      CALL printf
 /*         TEST_execute(led_test,  "LEDs", 1); */
0239B1:   3c14          MOV #1,T0
0239B3:   7a00020a      MOV #2 << #16,AC0
0239B7:   ec318e002c1e  AMAR *(#02c1eh),XAR0
0239BD:   7e3e1100      OR #03e11h,AC0,AC0
0239C1:   08ff7d        CALL TEST_execute
/*          printf("\n***ALL Tests Passed***\n"); */
0239C4:   ec31be002c24  AMAR *(#02c24h),XAR3
0239CA:   eb00b5        MOV XAR3,dbl(*SP(#00h))
0239CD:   6c023e64      CALL printf …
```

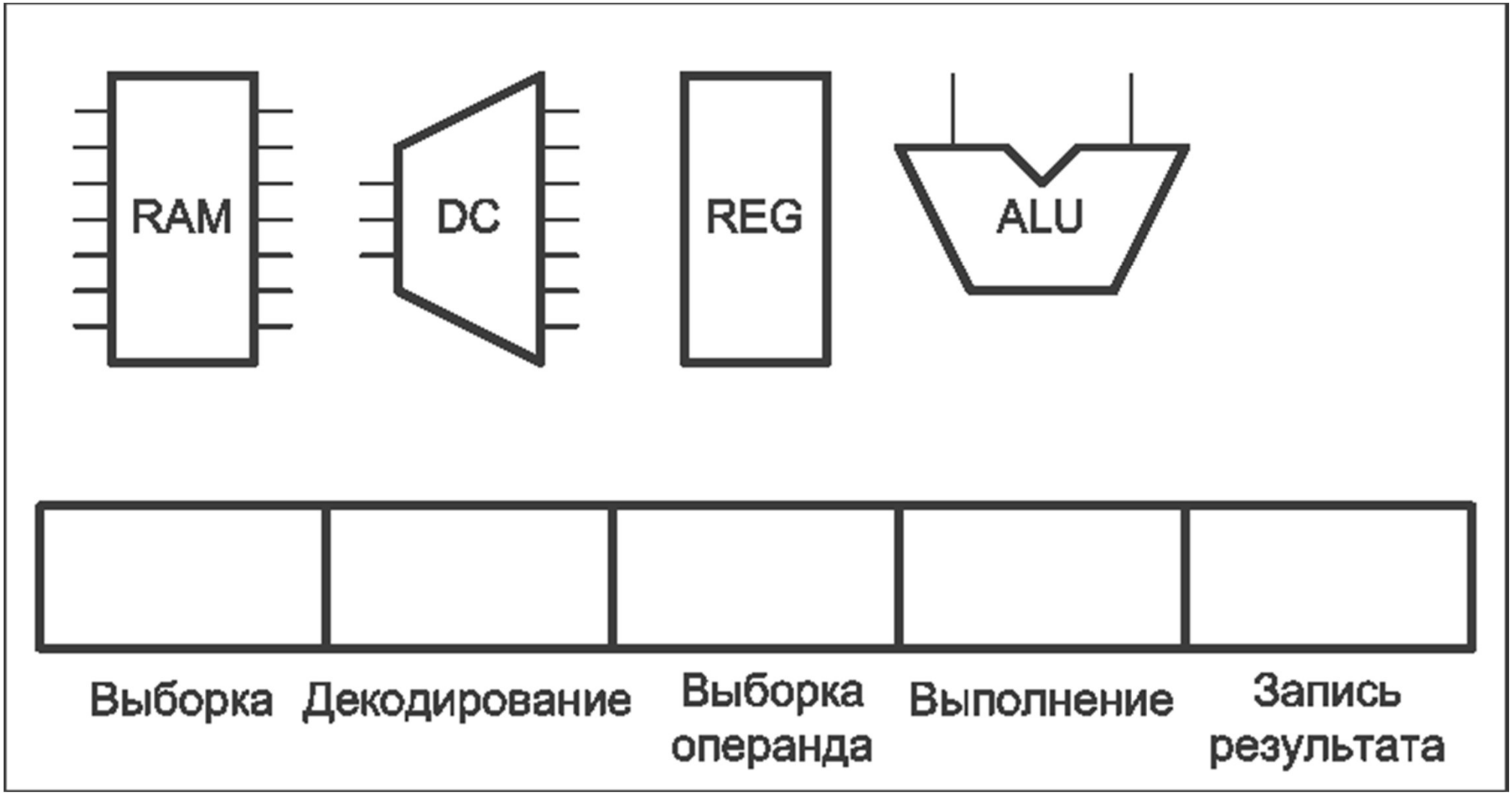


Декодер команд

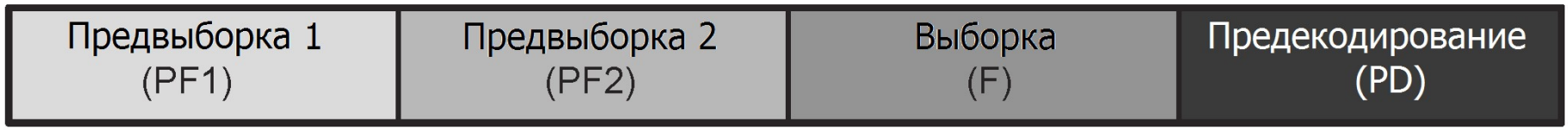


- Получение 6 байт кода для декодирования команды и определения ее размера.
- Передача команд другим устройствам.
- Выявление и декодирование команд, которые могут выполнятся параллельно.
- Передача устройствам непосредственных данных.
- Передача длины команды для сдвига очереди команд.

Конвейеризация

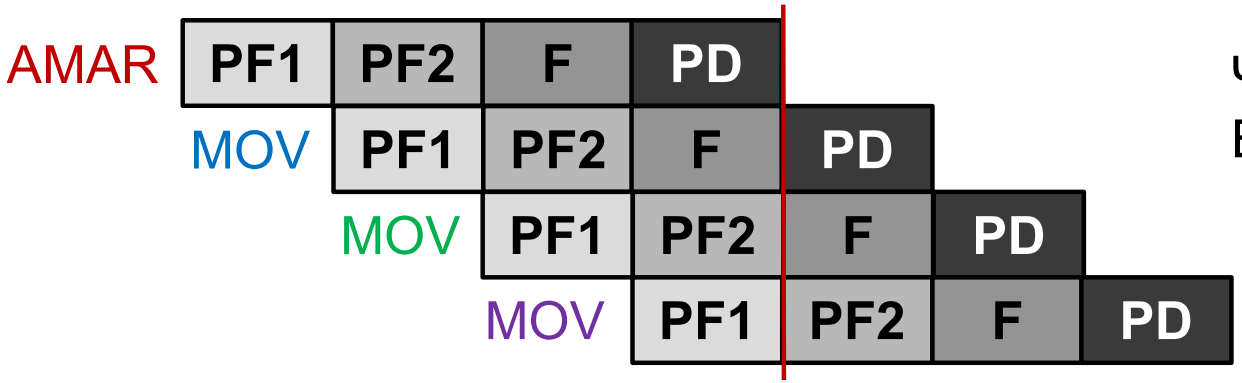


Конвейер выборки



- **PF1** – Prefetch 1
  (выдача адреса на шину PAB)
- **PF2** – Prefetch 2
  (ожидание устройства памяти)
- **F** – Fetch
  (получение данных и запись в очередь)
- **PD** – Predecode
  (предварительное декодирование команды в очереди: определение начала и окончания, обнаружение параллельности)

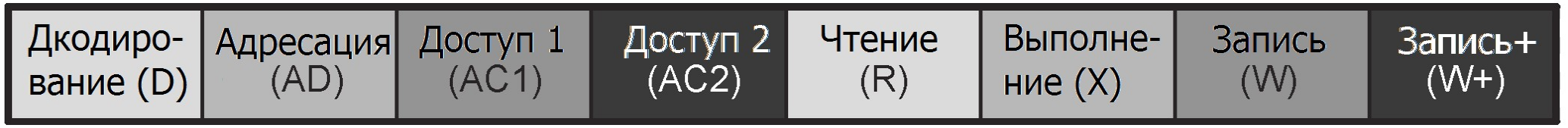
Конвейеризация выборки



Чтение из памяти – 4 такта
Выборка команды – 1 такт

```
/*          printf("\n%04x  T0", test);                  */
02399D:   ec31be002c14  AMAR *(#02c14h),XAR3     ; 100 %
0239A3:   a931002a49    MOV *(#02a49h),AR1       ; 75 %
0239A8:   eb00b5        MOV XAR3,dbl(*SP(#00h)) ; 50 %
0239AB:   c904          MOV AR1,*SP(#02h)        ; 25 %
0239AD:   6c023e64      CALL printf …            ; 0 %
```

Конвейер выполнения



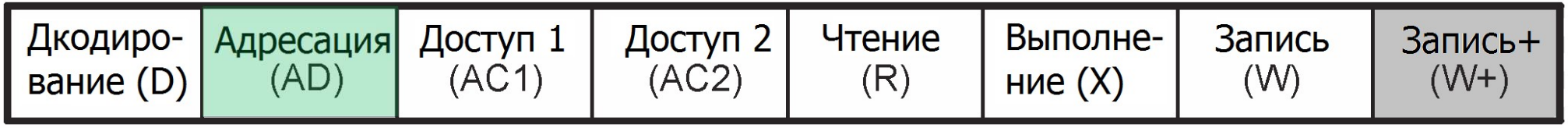
- **D** – Decode (декодирование команды)
- **AD** – Address (вычисление адреса)
- **AC1** – Access 1 (доступ 1)
- **AC2** – Access 2 (доступ 2)
- **R** – Read (чтение)
- **X** – Execute (выполнение)
- **W** – Write (запись в регистры)
- **W+** – Write (запись в память)

Декодирование команды (D)

Дкодиро-вание (D)	Адресация (AD)	Доступ 1 (AC1)	Доступ 2 (AC2)	Чтение (R)	Выполне-ние (X)	Запись (W)	Запись+ (W+)

- Чтение 6 байт из буфера команд
- Декодирование одной или двух команд
- Передача команд устройствам
- Чтение режимов адресации:
  CPL – режима адресации компилятора;
  ARMS – режим косвенной адресации;
  ARxLC, CDPLC – циклическая адресация.

Вычисление адреса(AD)



- Чтение регистров, используемых для адресации операндов
- Вычисление адреса операнда – выполнение операции ALU устройства адресации
- Выполнение префиксной операции адресации – **модификация регистров**
- Вычисление условий ветвления и **перезагрузка конвейера** при ветвлении, вызове подпрограммы, прерывании)

Доступ (AC1 и AC2)

Дкодиро-вание (D)	Адресация (AD)	Доступ 1 (AC1)	Доступ 2 (AC2)	Чтение (R)	Выполне-ние (X)	Запись (W)	Запись+ (W+)

- AC1 – передача адреса операнда на соответствующую шину адреса (BAB, CAB, DAB)

- AC2 – ожидание данных из памяти один такт

Чтение (R)

Дкодиро-вание (D)	Адресация (AD)	Доступ 1 (AC1)	Доступ 2 (AC2)	Чтение (R)	Выполне-ние (X)	Запись (W)	Запись+ (W+)

- Считывание данных из памяти или отображаемых в память регистров (магистрали данных BB, CB, DB)
- Чтение регистров устройства адресации, участвующих в операции обработки данных
- Вычисление условий для условных операций обработки данных

Выполнение (X)

Дкодиро-вание (D)	Адресация (AD)	Доступ 1 (AC1)	Доступ 2 (AC2)	Чтение (R)	Выполне-ние (X)	Запись (W)	Запись+ (W+)

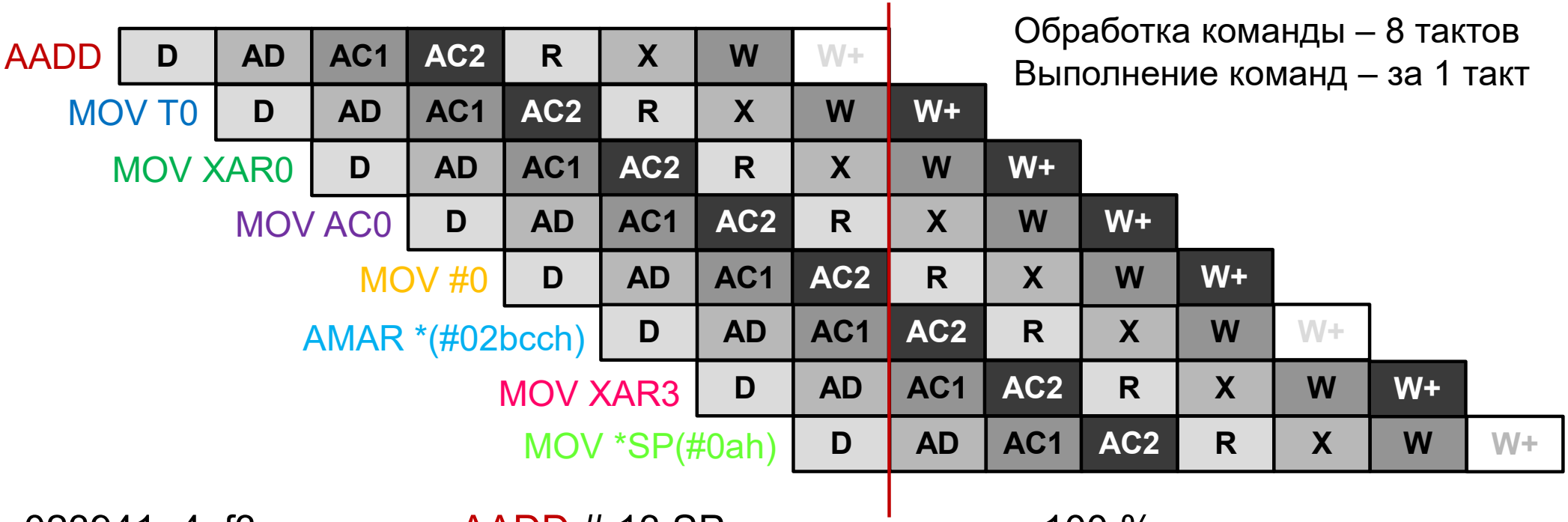
- Чтение и **модификация регистров**, участвующих в операции, кроме регистров, отображаемых в память (изменение регистров операционного устройства)
- Чтение и **модификация отдельных бит** (устройство битовых операций DB_BIT)
- Установка условий по результату операции обработки данных

Запись (W и W+)

Дкодиро-вание (D)	Адресация (AD)	Доступ 1 (AC1)	Доступ 2 (AC2)	Чтение (R)	Выполне-ние (X)	Запись (W)	Запись+ (W+)

- W – Запись данных в регистры, отображаемые в память, регистры периферийных устройств или начало записи данных в ячейку памяти
- W+ – Завершение записи данных в ячейку памяти

Конвейеризация выполнения



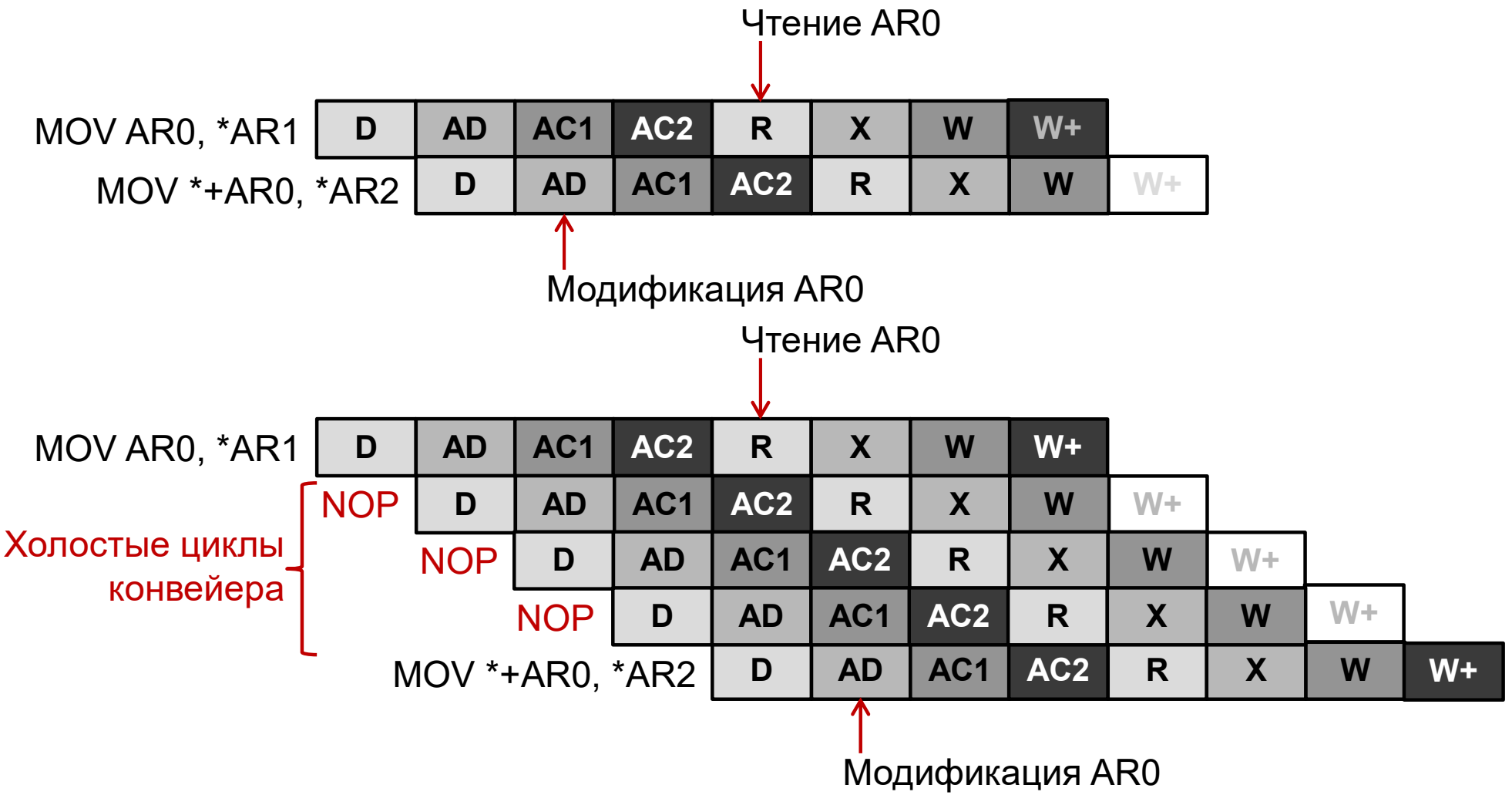
Обработка команды – 8 тактов
Выполнение команд – за 1 такт

```
023941: 4ef3          AADD #-13,SP                    ; 100 %
023943: c414          MOV T0,*SP(#0ah)                ; 87,5 %
023945: eb1085        MOV XAR0,dbl(*SP(#08h))         ; 75,0 %
023948: eb0c08        MOV AC0,dbl(*SP(#06h))          ; 62,5 %
02394B: e61600        MOV #0,*SP(#0bh)                ; 50,0 %
02394E: ec31be002bcc  AMAR *(#02bcch),XAR3            ; 37,5 %
023954: eb00b5        MOV XAR3,dbl(*SP(#00h))         ; 25,0 %
023957: a914          MOV *SP(#0ah),AR1               ; 12,5 %
023959: c904          MOV AR1,*SP(#02h)               ; 0 %
02395B: ed10bf        MOV dbl(*SP(#08h)),XAR3
02395E: eb08b5        MOV XAR3,dbl(*SP(#04h))
```

Перезагрузка конвейера

- Команды условных переходов **BCC**
- Команда безусловного перехода **B**
- Команда вызова подпрограммы **CALL**
- Программное прерывание **INTR**
- Внешние прерывания по сигналу **INT0**, **INT1**
- Внутренние немаскируемые прерывания по сигналу **NMI**
- Сброс процессора по сигналу **RESET** или командой программного сброса **RESET**

Конфликты записи до чтения

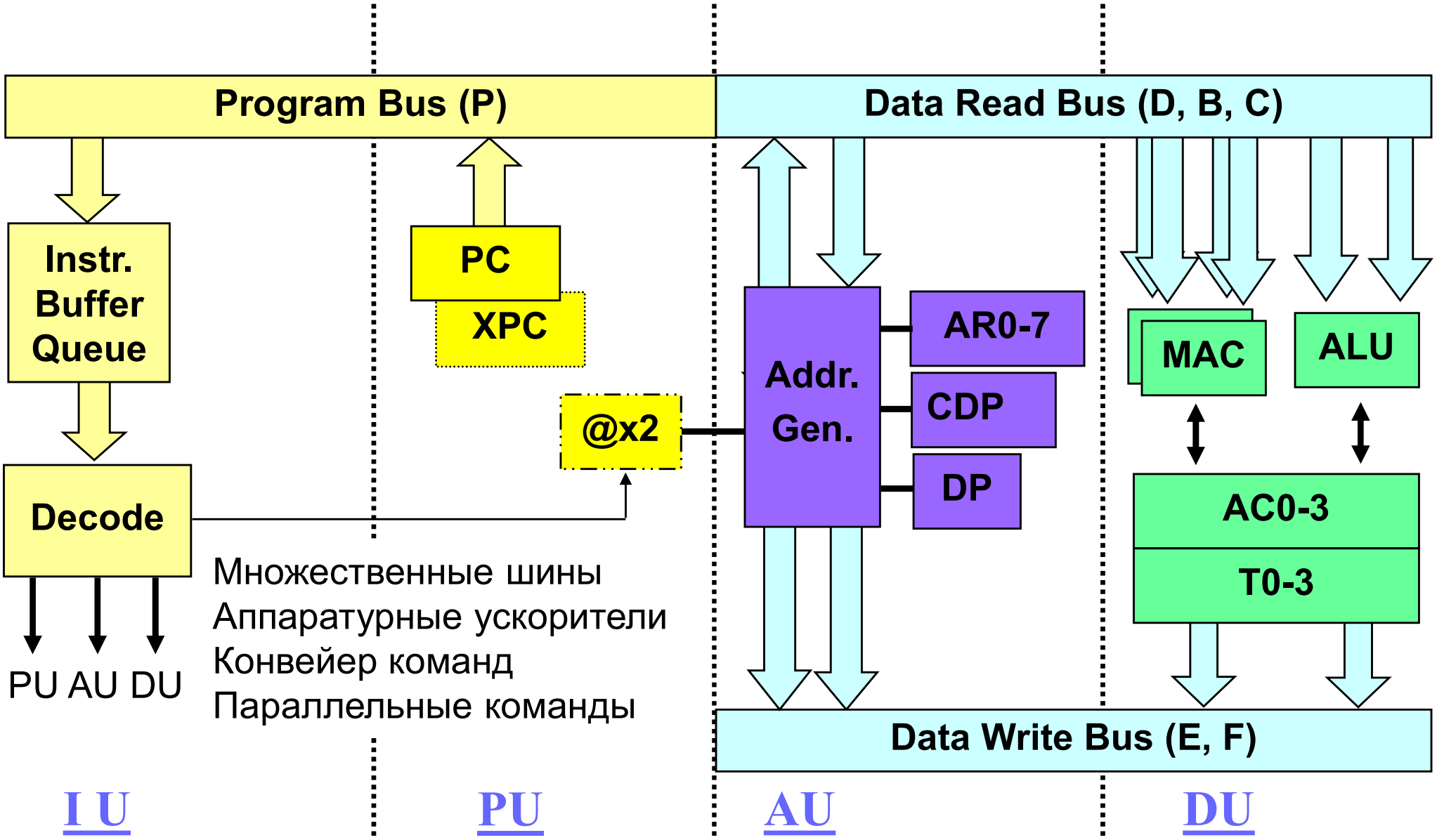


Уязвимость Meltdown

Конфликты доступа

- При параллельном выполнении команд забота о предотвращении конфликтов в конвейере ложится на компилятор (на программиста)

Независимые устройства



Виды параллелизма

- **Встроенный** (нет бита параллельности **E**)

```
MPY *AR0, *CDP, AC0      ; 1-й умножитель
:: MPY *AR1, *CDP, AC1   ; 2-й умножитель
```

- **Пользовательский** (бит **E** во 2-ой команде)

```
ADD AC0, AC1             ; АЛУ D-блока
|| XOR AR2, CDP          ; АЛУ A-блока, E=1
```

- **Комбинированный** (бит **E** в 3-ей команде)

```
MPY *AR3+, *CDP+, AC0  ; 1-й умножитель
:: MPY *AR4+, *CDP+, AC1 ; 2-й умножитель
|| MOV #5, AR1          ; РФ A-блока, E=1
```

- **Двойной доступ** (доступ к памяти, нет **E**)

```
MOV *AR-, AC1            ; CAB/BB, DAB/DB
MOV *CDP+, AC2           ; BAB/BB
```

Флаг параллельности

0010001E FSSSFDDD

RPTCC k8, cond
RETCC cond
BCC L8, cond
B L16
CALL L16
RPT k16
RPTB pmad
AND ACx << #SHIFTW[, ACy]
OR ACx << #SHIFTW[, ACy]
XOR ACx << #SHIFTW[, ACy]
ADD ACx << #SHIFTW, ACy
SUB ACx << #SHIFTW, ACy
SFTS ACx, #SHIFTW[, ACy]
SFTSC ACx, #SHIFTW[, ACy]
SFTL ACx, #SHIFTW[, ACy]
EXP ACx, Tx
MANT ACx, ACy
:: NEXP ACx, Tx
BCNT ACx, ACy,TCx, Tx
MAXDIFF ACx, ACy, ACz, ACw
DMAXDIFF ACx, ACy, ACz, ACw, TRNx
MINDIFF ACx, ACy, ACz, ACw
DMINDIFF ACx, ACy, ACz, ACw, TRNx
CMP[U] src RELOP dst, TCx
CMPAND[U] src RELOP dst, TCy, TCx
CMPAND[U] src RELOP dst, !TCy, TCx
CMPOR[U] src RELOP dst, TCy, TCx
CMPOR[U] src RELOP dst, !TCy, TCx

ROL BitOut, src, BitIn, dst
ROR BitIn, src, BitOut, dst
AADD TAx, TAy
AMOV TAx, TAy
ASUB TAx, TAy
AADD P8, TAx
AMOV P8, TAx
ASUB P8, TAx
AADD TAx, TAy
AMOV TAx, TAy
ASUB TAx, TAy
AADD P8, TAx
AMOV P8, TAx
ASUB P8, TAx
AADD XACsrc, XACdst
AMOV XACsrc, XACdst
ASUB XACsrc, XACdst
AADD XACsrc, XACdst
AMOV XACsrc, XACdst
ASUB XACsrc, XACdst
MOV k7, DPH
MOV k9, PDP
MOV k12, BK03
MOV k12, BK47
MOV k12, BKC
MOV k12, CSR
MOV k12, BRC0
MOV k12, BRC1

AND k8, src, dst
OR k8, src, dst
XOR k8, src, dst
MPYK[R] K8, [ACx,] ACy
MACK[R] Tx, K8, [ACx,] ACy
NOP
MOV src, dst
ADD [src,] dst
SUB [src,] dst
AND src, dst
OR src, dst
XOR src, dst
MAX [src,] dst
MIN [src,] dst
ABS [src,] dst
NEG [src,] dst
NOT [src,] dst
PSH src1, src2
POP dst1, dst2
MOV k4, dst
MOV –k4, dst
ADD k4, dst
SUB k4, dst
MOV HI(ACx), TAx
SFTS dst, #–1
SFTS dst, #1
MOV SP, TAx
MOV SSP, TAx

MOV src, dst

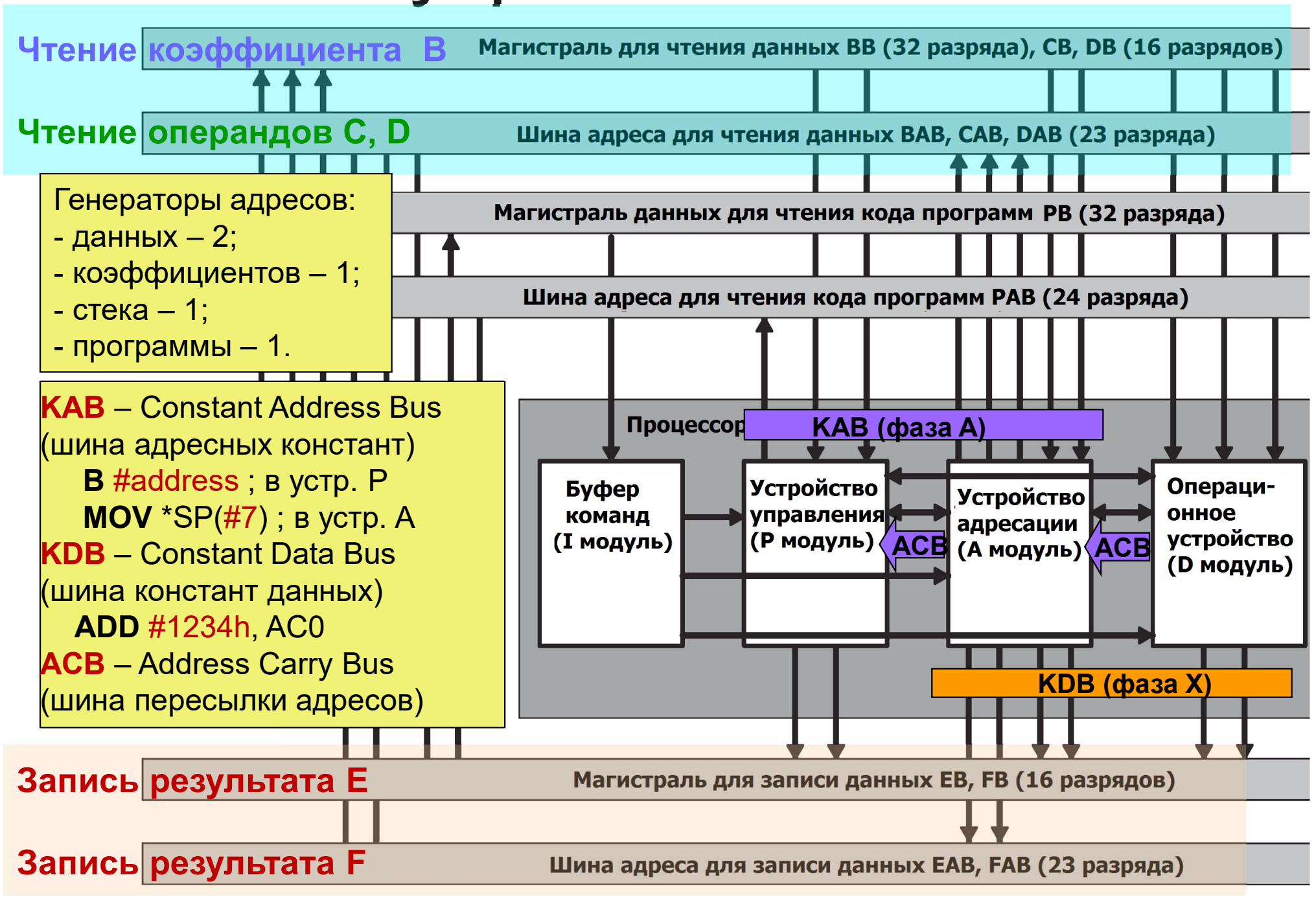
MOV CDP, TAx
MOV BRC0, TAx
MOV BRC1, TAx
MOV RPTC, TAx
BCLR k4, ST0_55
BSET k4, ST0_55
BCLR k4, ST1_55
BSET k4, ST1_55
BCLR k4, ST2_55
BSET k4, ST2_55
BCLR k4, ST3_55
BSET k4, ST3_55
RPT CSR
RPTADD CSR, TAx
RPTADD CSR, k4
RPTSUB CSR, k4
RET
B L7
RPTBLOCAL pmad
RPT k8
AADD K8,SP
SFTL dst, #1
SFTL dst, #–1
POP dst
POP dbl(ACx)
PSH src
PSH dbl(ACx)
POPBOTH xdst
PSHBOTH xsrc

MOV TAx, HI(ACx)
MOV TAx, SP
MOV TAx, SSP
MOV TAx, CDP
MOV TAx, CSR
MOV TAx, BRC1
MOV TAx, BRC0
ADD[R]V [ACx,] ACy
SQA[R] [ACx,] ACy
SQS[R] [ACx,] ACy
MPY[R] [ACx,] ACy
SQR[R] [ACx,] ACy
ROUND [ACx,] ACy
SAT[R] [ACx,] ACy
MAC[R] ACx, Tx, ACy[, ACy]
MAS[R] Tx, [ACx,] ACy
MPY[R] Tx, [ACx,] ACy
MAC[R] ACy, Tx, ACx, ACy
ADD ACx << Tx, ACy
SUB ACx << Tx, ACy
SFTCC ACx, TCx
SFTL ACx, Tx[, ACy]
SFTS ACx, Tx[, ACy]
SFTSC ACx, Tx[, ACy]
SWAP ( )

Условия параллелизма

- Суммарная длина двух команд не более 6 байт;
- Задано параллельное выполнение двух команд (например, E бит в коде первой или второй команды);
- Нет конфликта ресурсов (шин, регистров, устройств, регистров, ячеек памяти), используемых двумя командами;
- Нет команд, запрещенных для параллельного выполнения (INTR, TRAP, RESET).

Внутренние шины



Правила распараллеливания 1

- Нельзя распараллелить команды с непосредственными данными длиной 16 бит и более (увеличивают длину команд)
- Нельзя распараллеливать команды ветвления с полным адресом перехода P24 (BCC, CALLCC), команды прерывания (INTR, TRAP) и команду сброса (RESET)
- Ограничено использование квалификаторов команд (xxx.CR –циклическая адресация, xxx.LR – линейная адресация)

Правила распараллеливания 2

- В одной фазе конвейера не может выполняться чтение и запись одной и той же ячейки или регистра.
- Команды модификации указателя стека не могут выполняться вместе с другими командами, его изменяющими.
- Могут распараллеливаться только те команды, которые реализуются различными устройствами: A, D, P.
- Не могут распараллеливаться две команды, реализуемые P-блоком

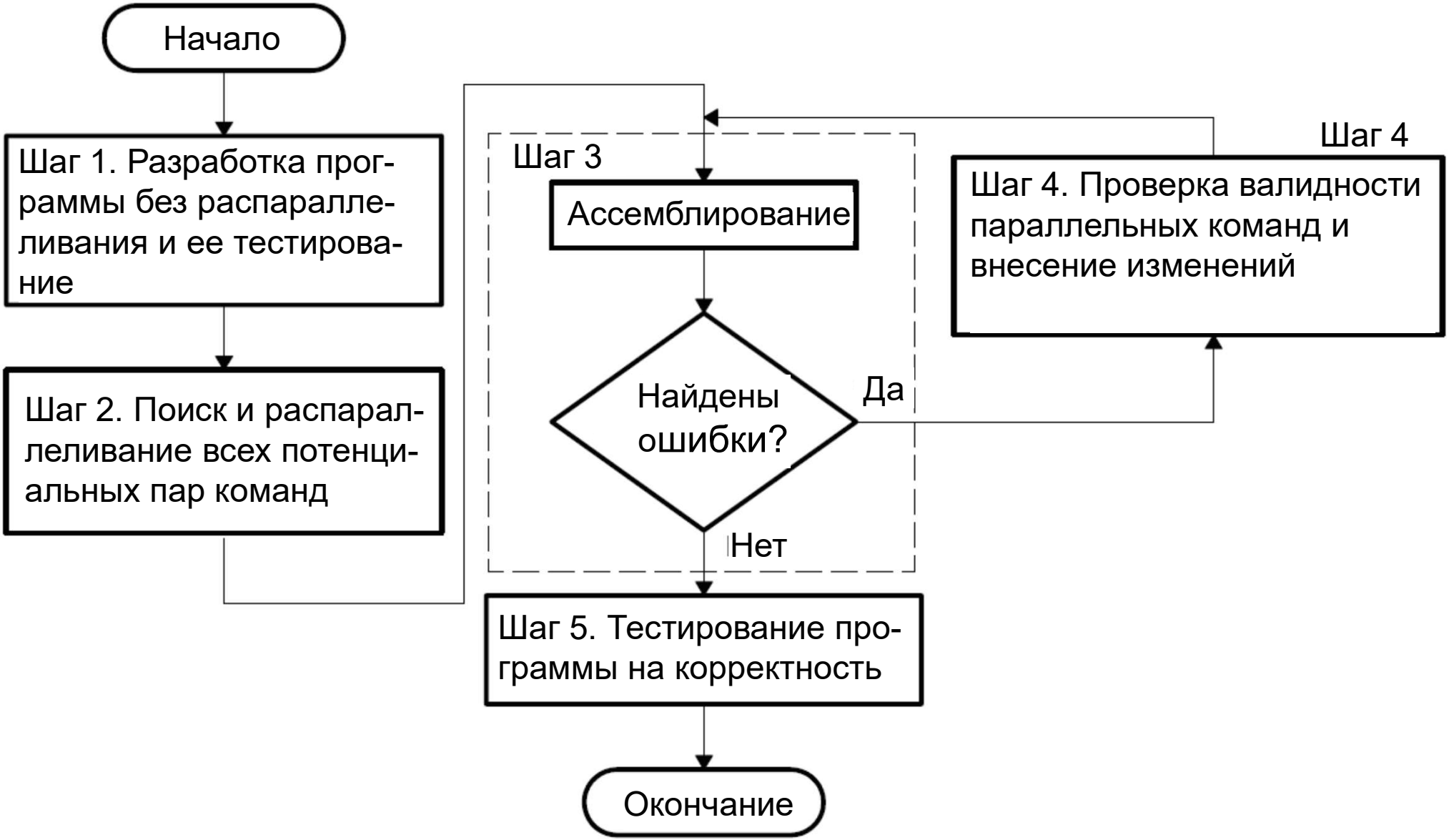
Правила распараллеливания 3

- Могут распараллеливаться команды, которые используют различные блоки устройства D (сдвигатель, два умножителя, АЛУ)
- Не распараллеливаются обращения в пространство ввода-вывода (port) или к отображаемым в память регистрам (mmap)

Параллельные операции

	A-unit ALU	A-unit Swap	A-unit Load	A-unit Store	D-unit ALU	D-unit Shifter	D-unit MAC	D-unit Load	D-unit Store	D-unit Shift, Store	D-unit Swap	P-unit Control	P-unit Load	P-unit Store
A-unit ALU	X													
A-unit Swap		X												
A-unit Load														
A-unit Store														
D-unit ALU					X	X	X							
D-unit Shifter					X	X	X			X				
D-unit MAC					X	X	X							
D-unit Load														
D-unit Store														
D-unit Shift, Store						X				X				
D-unit Swap											X			
P-unit Control												X		
P-unit Load														
P-unit Store														

Процесс распараллеливания



Примеры распараллеливания 1

- Чтение и запись регистров
  MOV *AR2, AC1
  || MOV BRC0, *AR3
- Операции устройств A и D
  ADD T0, AR1
  || MOV HI(saturate(AC1<<1)), *AR2
- Операции различных блоков устройства D
  MOV HI(saturate(AC1<<1)), *AR2
  || ADD AC1, AC2

Примеры распараллеливания 2

- Операции устройств P и D
  || MAC *AR1+, *AR3, AC0
  RTPLOCAL label
- Команды с внутренней параллельностью
  ADD dual(*AR2), AC0, AC1
  || MOV AC2, dbl(*AR6) ; dbl – 32 бита (EB,FB)
  ; dual – двойной доступ к памяти DARAM
- Команды ветвления
  XCC label, T0 == #0 ; ветвление и конвейере
  || MOV #0, AR0
- label:

Трассировка данных