Организация ЭВМ и вычислительных систем

ЛЕКЦИЯ 11

Тема 4. Центральный процессор

4.7. Структура центрального процессора

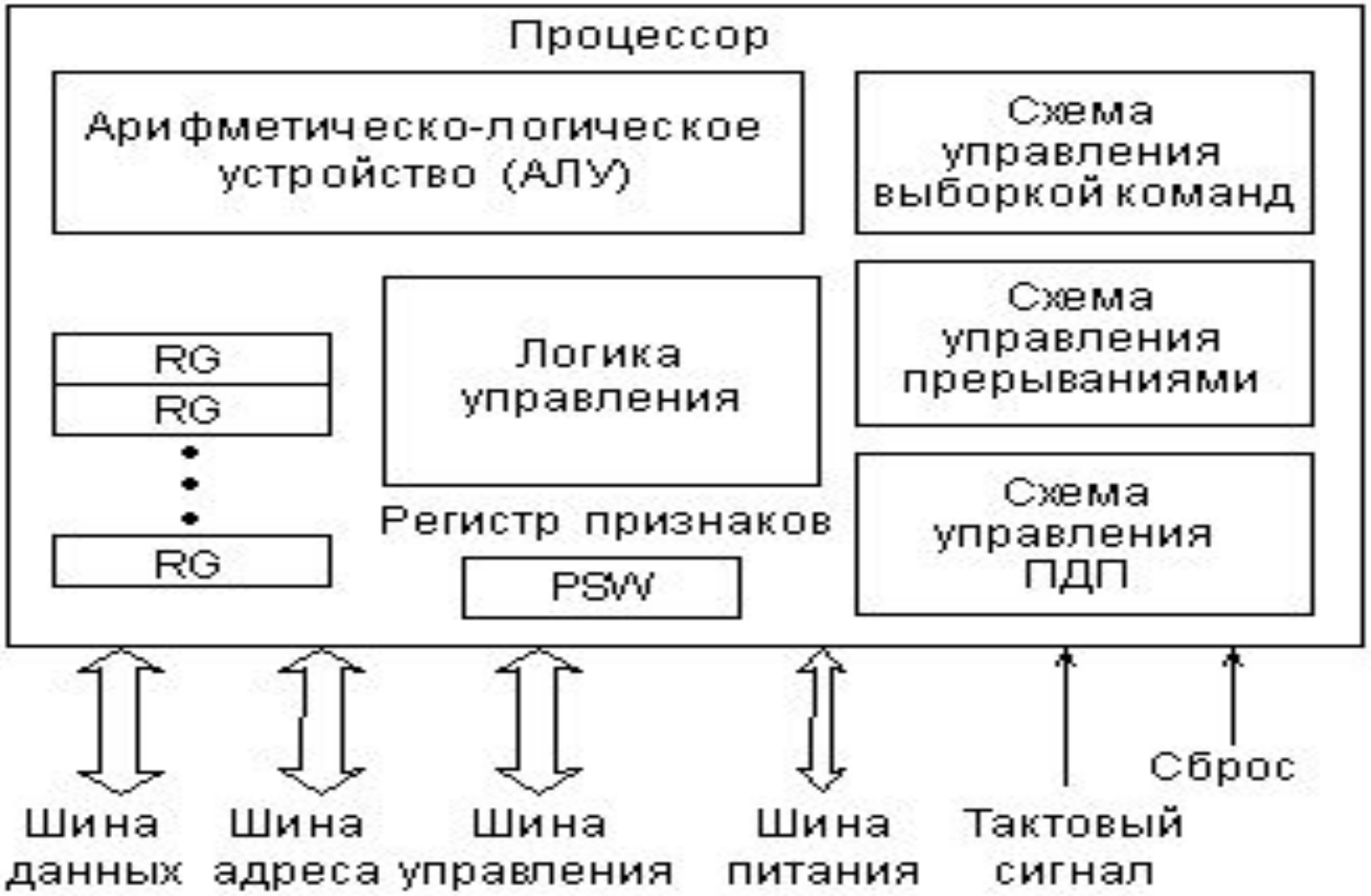


Схема управления выборкой команд выполняет чтение команд из памяти и их дешифрацию. Чтение команд производится последовательно, одна за другой, так как не возможно одновременное выполнение предыдущей команды и выборка следующей команды.

Арифметико-логическое устройство предназначено для обработки информации в соответствии с полученной процессором командой: логической или арифметические операциями. Над какими кодами производится операция и куда помещается ее результат, определяет выполняемая команда. Быстродействие АЛУ во многом определяет производительность процессора. Для повышения производительности разработчики стремятся довести время выполнения команды до одного такта, а также обеспечить работу АЛУ на возможно более высокой частоте.

Операции над числами с плавающей точкой и другие сложные операций выполняются с помощью математического сопроцессора, который заменяли основной процессор на время выполнения таких

Регистры процессора

Регистры процессора представляют собой ячейки очень быстрой памяти и служат для временного хранения различных кодов: данных, адресов, служебных кодов. Операции с этими кодами выполняются предельно быстро, поэтому, чем больше внутренних регистров, тем лучше. На быстродействие процессора сильно влияет разрядность регистров. Именно разрядность регистров и АЛУ называется ***внутренней разрядностью процессора***, которая может не совпадать с внешней разрядностью.

По отношению к назначению внутренних регистров существует два основных подхода. Первый: ***каждому регистру отводит строго определенную функцию.***

Второй подход состоит в том, чтобы ***все (или почти все) регистры сделать равноправными.***

При первом подходе упрощается организация процессора и уменьшается время выполнения команды; с другой – снижает гибкость, а иногда и замедляет работу программы. Так, некоторые арифметические операции и обмен с устройствами ввода/вывода проводятся только через один регистр – ***аккумулятор***, в результате чего при выполнении таких процедур потребуется несколько дополнительных пересылок между регистрами.

При втором подходе достигается высокая гибкость, но необходимо усложнение структуры процессора.

Существуют и промежуточные решения, когда половина регистров используется для данных, и они являются взаимозаменяемыми, а другая половина – для адресов, и они также взаимозаменяемы.

В первую группу входят ***регистры общего назначения***. В процессорах имеются восемь 32-битовых регистров общего назначения ***EAX***, ***EBX***, ***ECX, EDX***, ***ESI, EDI***, ***EBP,*** и ***ESP***. Процессоры могут обращаться к 16-битовым половинам 32-битовых регистров. При необходимости возможна работа с половинами регистров, поскольку они разделены на старшую и младшую половину, называемые ***AH*** и ***AL***, ***BH*** и ***BL*** и так далее. Такое разделение регистров имеется во всех процессорах. Значительная часть внутренних операций компьютеров производится с использованием регистров общего назначения.

Следующая группа из шести регистров помогает процессору обращаться к памяти. Они называются ***сегментными регистрами*** и каждый из них помогает обращаться к области (или сегменту) памяти. В современных процессорах длина сегмента переменная и варьируется от одного байта до 4 Гбайт.

Регистр ***CS сегмента кода программы*** показывает, в каком месте памяти находится программа. Регистр ***DS сегмента данных*** локализует используемые программой данные. Регистр ***ES дополнительного сегмента*** дополняет сегмент данных. Регистр ***SS сегмента стека определяет стек компьютера***. Еще два ***сегментных регистра FS*** и ***GS предназначенны для адресации памяти***

Последняя группа регистров используется совместно с сегментным регистром для локализации в памяти конкретных байтов. ***Регистр указателя команды (счетчик команд) IP определяет точку, где выполняется программа***. ***Регистры указателя стека SP*** и ***указателя базы BP помогают следить за информацией в стеке*** (***стек — это область памяти, где хранится информация о текущих действиях компьютера***).

Регистр признаков (***регистр состояния***) занимает особое место, хотя он также является внутренним регистром процессора. Содержащаяся в нем информация – это ***состояния процессора*** (***PSW – Processor Status Word***). ***Каждый бит этого слова*** (***флаг***) ***содержит информацию о результате предыдущей команды***

Бит нулевого результата устанавливается в том случае, когда результат выполнения предыдущей команды — нуль, и очищается в том случае, когда результат выполнения команды отличен от нуля. Эти биты (***флаги***) используются командами условных переходов, например, командой перехода в случае нулевого результата. В этом же регистре иногда содержатся флаги управления, определяющие режим выполнения некоторых команд.

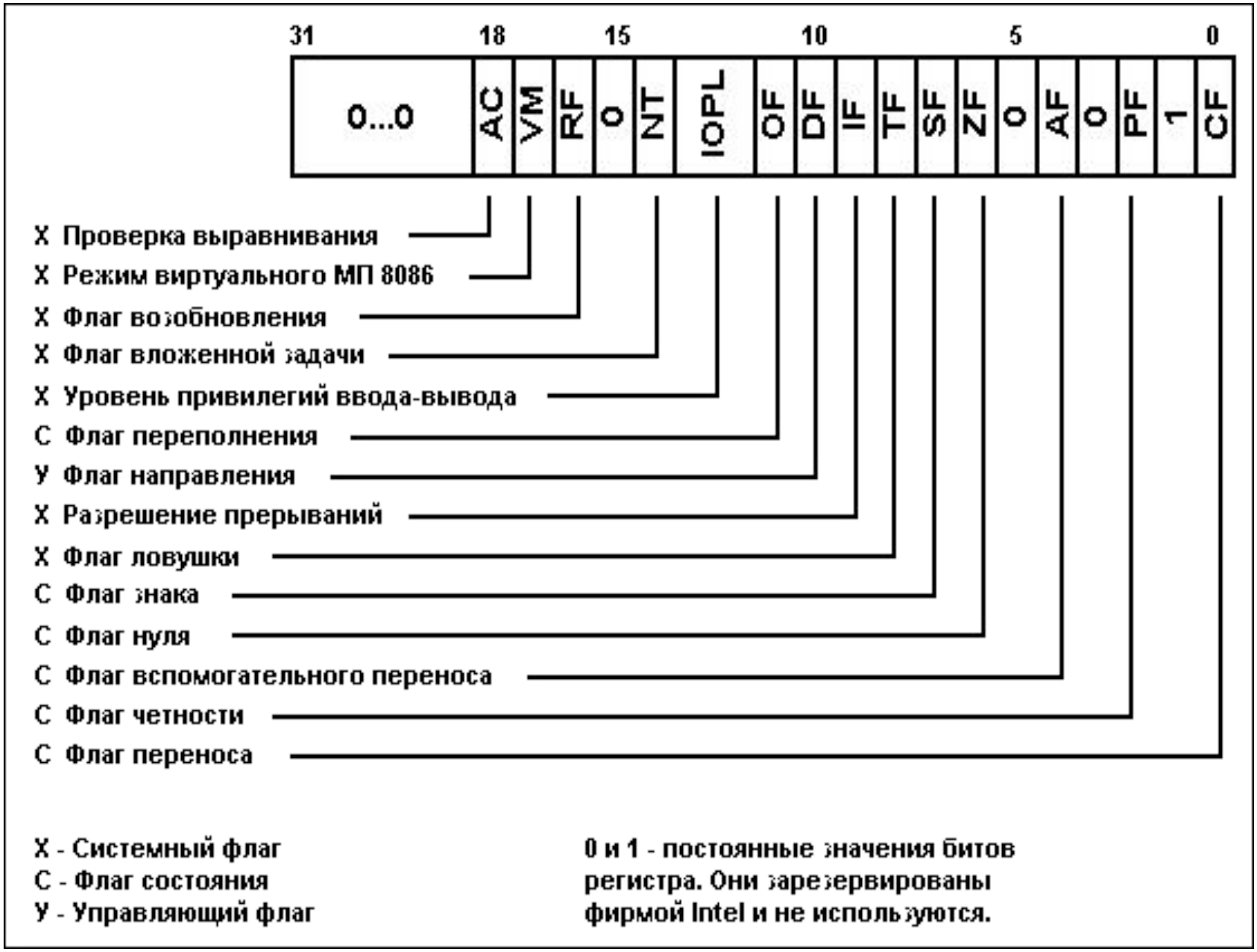

31
18
15
10
5
0
0...0
AC
VM
RF
0
NT
IOPL
OF
DF
IF
TF
SF
ZF
0
AF
0
PF
1
CF
X Проверка выравнивания
X Режим виртуального МП 8086
X Флаг возобновления
X Флаг вложенной задачи
X Уровень привилегий ввода-вывода
C Флаг переполнения
У Флаг направления
X Разрешение прерываний
X Флаг ловушки
C Флаг знака
C Флаг нуля
C Флаг вспомогательного переноса
C Флаг четности
C Флаг переноса
X - Системный флаг
C - Флаг состояния
У - Управляющий флаг
0 и 1 - постоянные значения битов регистра. Они зарезервированы фирмой Intel и не используются.

Схема управления прерываниями обрабатывает поступающий в процессор запрос прерывания, определяет адрес начала программы обработки прерывания, обеспечивает переход к этой программе после выполнения текущей команды и сохранения в памяти текущего состояния регистров процессора. По окончанию программы обработки прерывания процессор возвращается к прерванной программе с восстановленными из памяти значениями внутренних регистров.

Схема управления прямым доступом к памяти служит для временного отключения процессора от внешних шин и приостановки работы процессора на время предоставления прямого доступа запросившему его устройству.

Логика управления организует взаимодействие всех узлов процессора, перенаправляет данные, синхронизирует работу процессора с внешними сигналами, а также реализует процедуры ввода и вывода информации.

Таким образом, схема выборки команд выбирает последовательно команды из памяти, затем эти команды выполняются. На входы АЛУ подаются обрабатываемые данные из памяти или из внутренних регистров. Во внутренних регистрах хранятся коды адресов данных, расположенных в памяти. Результат обработки в АЛУ заносится во внутренний регистр или в память. Информация может переписываться из памяти или из устройства ввода/вывода во внутренний регистр или из внутреннего регистра в память. Внутренние регистры любого процессора обязательно выполняют ***две служебные функции***:

- ***определяют адрес в памяти, где находится выполняемая в данный момент команда (функция счетчика команд или указателя команд);***
- ***определяют текущий адрес стека (функция указателя стека).***

Содержимое ***указателя (счетчика) команд*** изменяется следующим образом. При включении питания в него заносится первый адрес программы начального запуска. После выборки из памяти каждой следующей команды значение указателя команд автоматически увеличивается на единицу – следующая команда будет выбираться из следующего по порядку адреса памяти. При выполнении команд перехода, нарушающих последовательный перебор адресов памяти, в указатель команд принудительно записывается новое значение, начиная с которого адреса команд опять же будут перебираться

4.8. Обработка команды

Фаза ***извлечения команды*** включает в себя операции ***определения адреса команды, выборки адреса команды, выборки команды***, а фаза ***выполнения*** – ***вычисление адресов операндов, выборки операндов, исполнения операции, записи результата***. После завершения выполнения команды компьютер выполняет следующую команду и так далее, пока не встретится команда остановки.

1. Определение адреса команды.

Адрес команды хранится в регистре-счетчике команд и, в случае линейного выполнения программы, после выполнения каждой команды счетчик команд увеличивает содержимое на количество слов команды. В случае безусловного перехода в счетчик команд записывается адрес перехода.

2. Выборка адреса команды.

Для чтения команды из оперативной памяти процессор устанавливает адрес команды на шине адреса и производит выборку.

3. Выборка команды.

Блок сопряжения выполняет ввод команды через итерфейс с памятью. Команда запоминается в КЭШ команд.

4. Дешифрация команды.

Если команда состоит из нескольких слов, то в дешифратор кода команды передается только первое слово команды, которое содержит код операции и признаки адресации. По первому слову определяется длина команды и выбор следующих слов происходит по мере необходимости. Процесс дешифрации может быть разделен на первичную и вторичную. Первичная дешифрация определяет тип команды, то есть группу, к которой команда относится. Первичная дешифрация позволяет уменьшить объем алгоритма обработки программ за счет одинаковой обработки команд одного типа. Вторичная дешифрация выполняется на более поздних этапах, после вычисления адресов операндов. Для команд арифметико-логической группы вторичная дешифрация может выполняться прямо в АЛУ.

5. Вычисление адресов операндов.

Если команда адресная, то на этом этапе вычисляются адреса операндов. Вычисление адреса и выборка для каждого операнда чередуются. Адрес операнда, если он является адресом ячейки ОЗУ, помещается в регистр адреса памяти.

6. Выборка операндов.

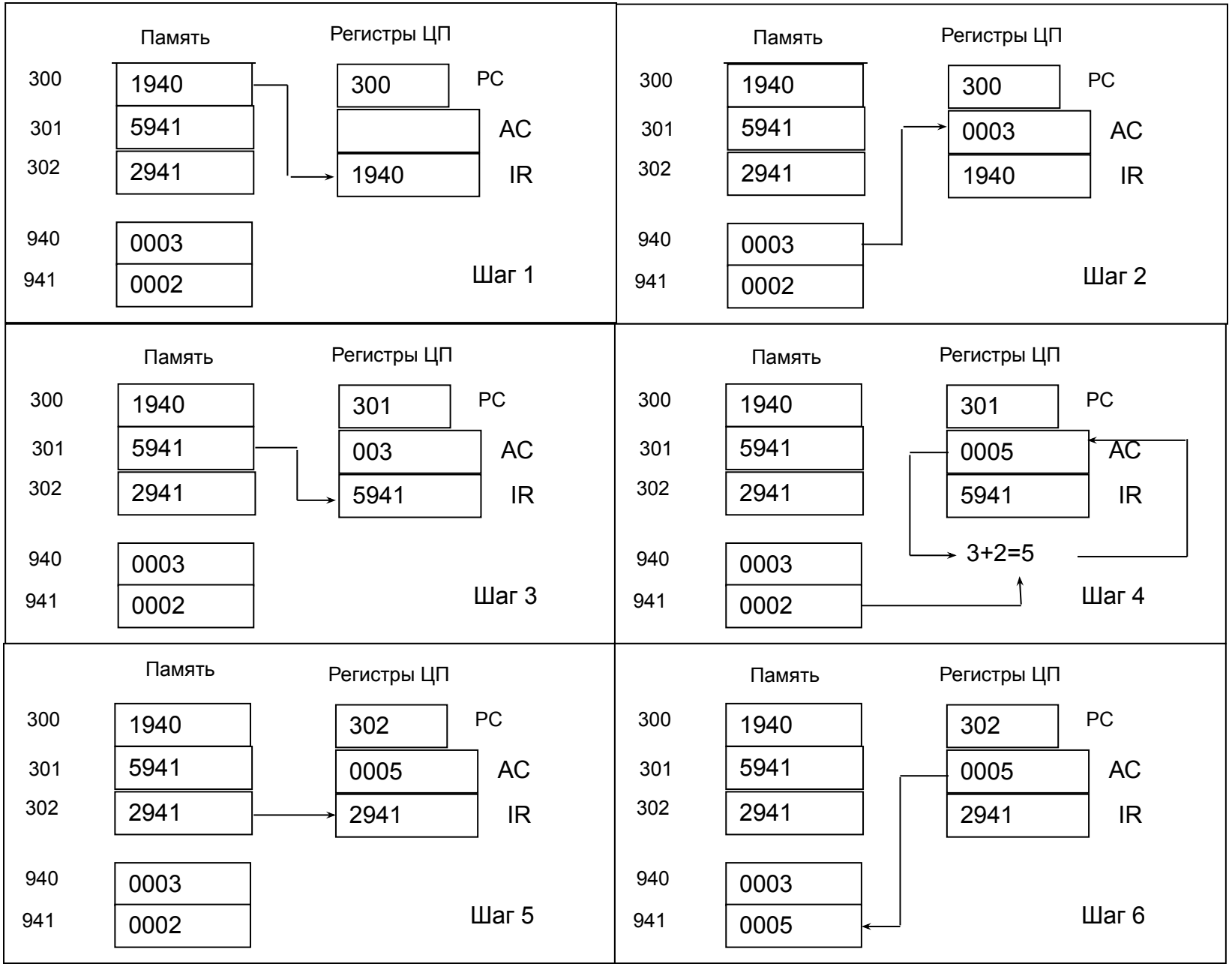
Выборка операндов производится для большинства адресных команд арифметико-логической группы. Содержимое ячейки памяти вводится в процессор для выполнения операции в АЛУ процессора. Если операнды размещаются во внутренних регистрах процессора, то операция выполняется значительно быстрее, чем при извлечении данных из памяти.

7. Исполнение операции.

На этой стадии производится вторичная дешифрация команды непосредственно в АЛУ, где и выполняется операция над подготовленными заранее операндами. Кроме арифметических или логических операций могут выполняться операции по пересылке операндов; в этом случае операнд извлекается из соответствующего регистра и пересылается на место операнда-приемника. Если выполняется команда безусловного перехода, то вычисленный адрес перехода записывается в регистр-счетчик команд. Команды вызова подпрограмм требуют запоминания состояния вычислительного процесса. Для этого используется сохранение данных в стеке – области памяти, предназначенной для записи данных в определенной последовательности и их последовательного извлечения. Для записи данных в стек и их извлечения из стека используется ***специальный адресный регистр, изменяющий адрес***. Указатель стека всегда указывает на следующую ячейку памяти.

8. Запись результата.

После выполнения команды результат операции обычно помещается в регистр аккумулятор. Затем он должен быть записан в оперативную память и, если это необходимо, выведен на внешнее запоминающее устройство, на дисплей монитора или передано другому внешнему устройству. Ввод и вывод информации для освобождения центрального процессора производят специальные каналы ввода/вывода. При этом канал управляется процессором ввода/вывода, который анализирует ситуацию и осуществляет обмен.


Память
Регистры ЦП
300
1940
301
5941
302
2941
940
0003
941
0002
300
PC
AC
IR
1940
Шаг 1
Память
Регистры ЦП
300
1940
301
5941
302
2941
940
0003
941
0002
300
PC
0003
AC
1940
IR
Шаг 2
Память
Регистры ЦП
300
1940
301
5941
302
2941
940
0003
941
0002
301
PC
003
AC
5941
IR
Шаг 3
Память
Регистры ЦП
300
1940
301
5941
302
2941
940
0003
941
0002
301
PC
0005
AC
5941
IR
3+2=5
Шаг 4
Память
Регистры ЦП
300
1940
301
5941
302
2941
940
0003
941
0002
302
PC
0005
AC
2941
IR
Шаг 5
Память
Регистры ЦП
300
1940
301
5941
302
2941
940
0003
941
0005
302
PC
0005
AC
2941
IR
Шаг 6

Адрес ***300*** первой команды, хранится в программном ***регистре-счетчике PC***. Эта команда. представленная шестнадцатеричным числом ***1940***. загружается в ***регистр команд IR***, а показание счетчика увеличивается на ***1***.

Первые 4 бит (первая шестнадцатеричная цифра – ***1***) регистра команд указывают на то, что нужно загрузить значение в ***аккумулятор AC***. Остальные 12 бит (три шестнадцатеричные цифры) указывают адрес ячейки памяти ***940***, из которой будут загружаться данные.

Из ячейки ***301*** извлекается следующая команда ***5941***; значение программного счетчика увеличивается на ***1***.

К содержимому аккумулятора прибавляется содержимое яейки ***941***, и результат снова заносится в аккумулятор.

Из ячейки ***302*** извлекается следующая команда ***5941***, затем значение программного счетчика увеличивается на 1.

Содержимое аккумулятора заносится в ячейку ***941***.

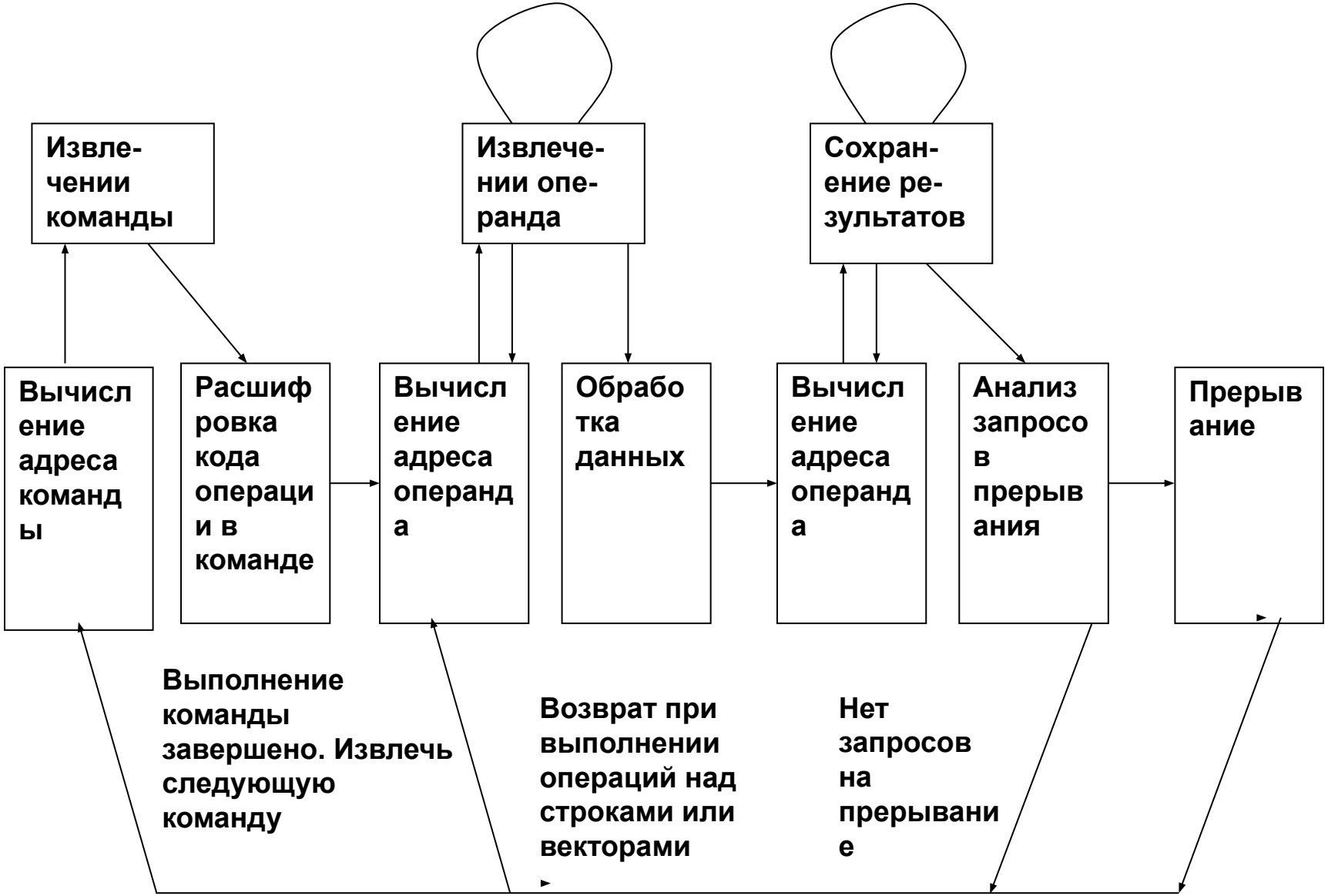
Этот пример показывает, что для сложения содержимого ячеек 940 и 941 необходимы три цикла команды. При более сложном наборе команд циклов понадобилось бы

4.9. Конвейерная обработка команд

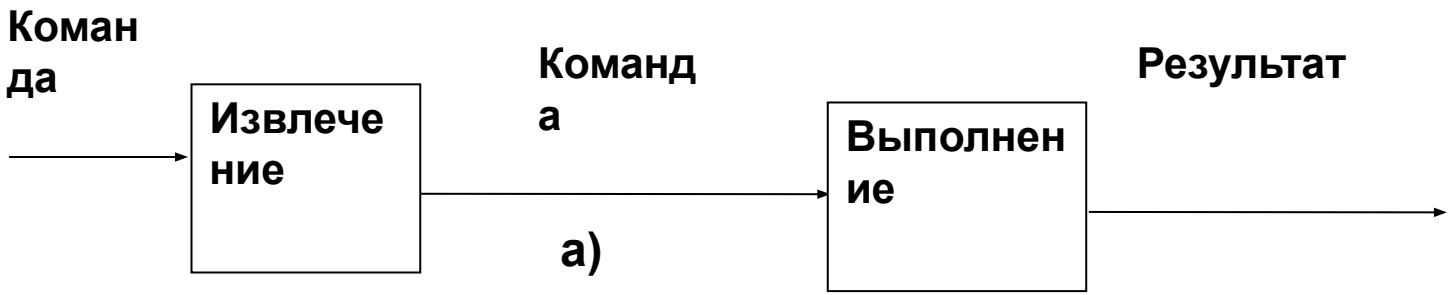
В последнее время широкое распространение получило такое направление в организации процесса обработки команд, как ***конвейерная обработка***.

Конвейерная обработка команд в вычислительных системах напоминает сборку изделия на конвейере в машиностроении. Эффективность сборочного конвейера основана на том, что изделие последовательно проходит через определенные рабочие позиции, причем операция на всех позициях выполняется одновременно.

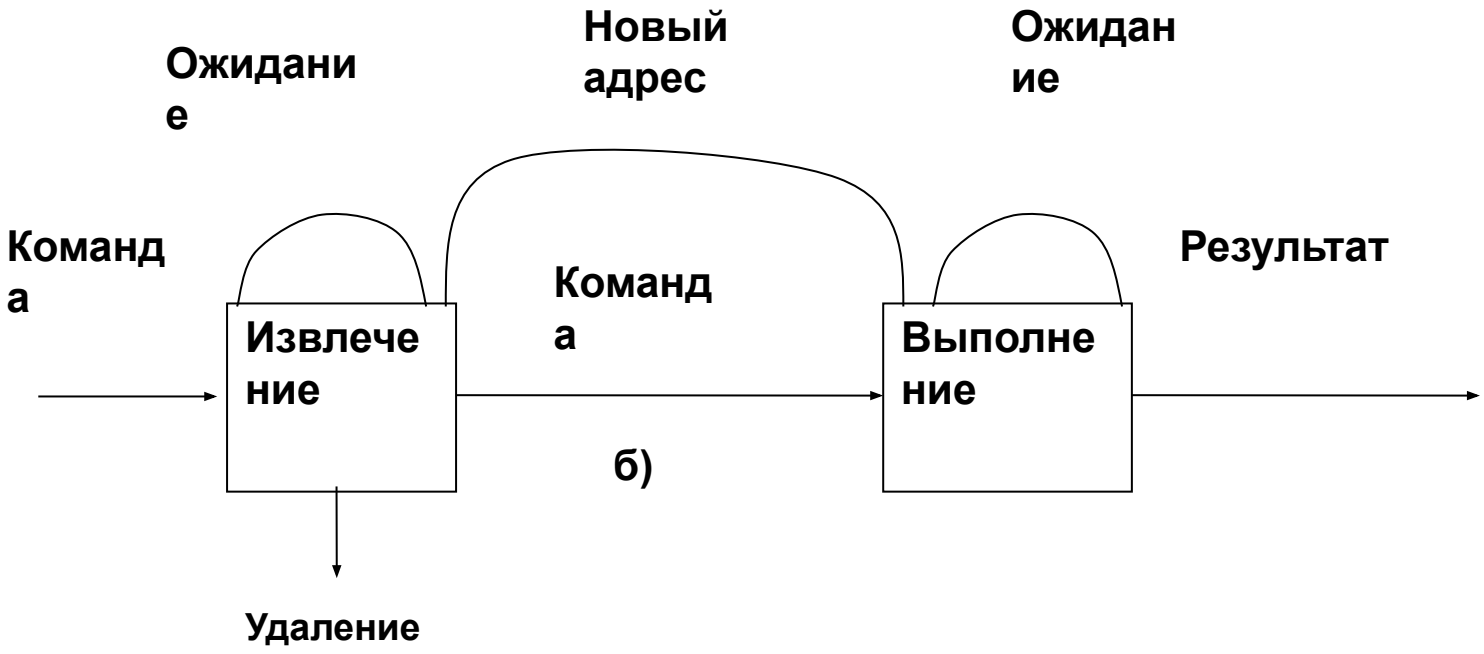
При реализации конвейера применительно к выполнению машинных команд процесс обработки распадается на множество этапов. Так, последовательность обработки основного цикла предусматривает разбивку на 10 подзадач, выполняемых последовательно. При такой разбивке имеются предпосылки для совмещения выполнения отдельных операций во времени


Извле-
чении
команды
Извлече-
нии опе-
ранда
Сохран-
ение ре-
зультатов
Вычисл
ение
адреса
команд
ы
Расшиф
ровка
кода
операци
и в
команде
Вычисл
ение
адреса
операнд
а
Обрабо
тка
данных
Вычисл
ение
адреса
операнд
а
Анализ
запросо
в
прерыв
ания
Прерыв
ание
Выполнение команды завершено. Извлечь следующую команду
Возврат при выполнении операций над строками или векторами
Нет запросов на прерывание

Пусть процесс обработки команды разделяется на две фазы. При выполнении команды существуют интервалы времени, когда обращение к памяти не производится. Эти интервалы можно использовать для извлечения следующей команды параллельно с выполнением текущей. Конвейер имеет две независимые “рабочие позиции” – извлечение команды и выполнение команды. На первой позиции команда извлекается из памяти и загружается в буфер. Когда вторая позиция будет свободной, первая передаст ей команду из буфера. Пока команда будет выполняться на второй позиции, на первой позиции можно использовать любой свободный цикл обращения к памяти и извлечь следующую команду, загрузив ее в буфер. Этот процесс называется ***извлечением команды с опережением или наложением извлечения***. Такая организация ускорит обработку команды. Если оба процесса реализуются за одно и то же время, то цикл обработки команды сократится вдвое


Команда
Извлечение
Команда
Выполнение
Результат

а)

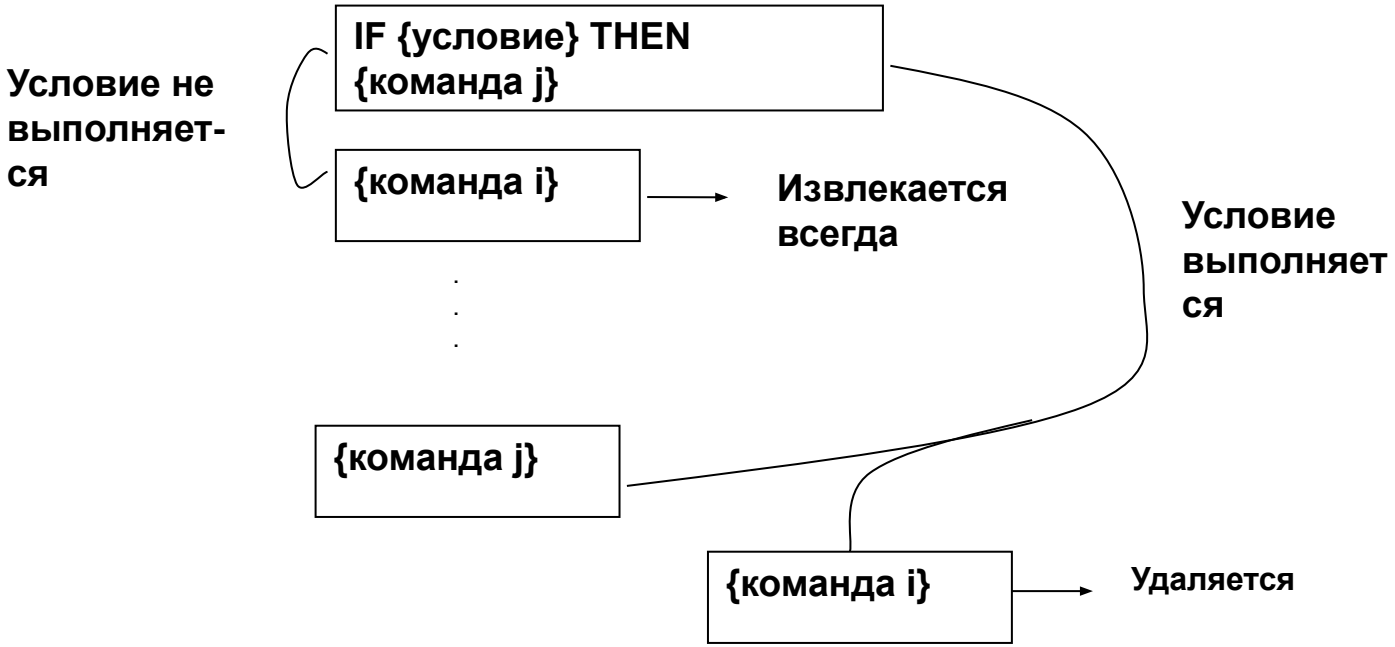

Ожидание
Новый адрес
Ожидание
Команда
Извлечение
Команда
Выполнение
Результат
Удаление

б)

На деле возможность увеличения скорости обработки вдвое на практике не реализуется. Причина этого – следующие :

1. Время выполнения команды в общем случае больше времени извлечения. Выполнение команды может потребовать извлечения и сохранения в памяти значений операндов и выполнения определенных операций с ними. Следовательно, «позиция извлечения» конвейера будет простаивать, дожидаясь, пока заполненный ею буфер не освободится.
2. При выполнении команды условного перехода, нельзя предсказать, каков будет адрес следующей команды. Следовательно, позиция извлечения должна дожидаться завершения выполнения предыдущей команды, получить от нее адрес следующей команды и после этого приступать к ее извлечению.

Потерю времени по второй причине можно сократить, если по определенному правилу предположить заранее результат условного перехода. Самое простое правило: «После извлечения команды условного перехода всегда извлекать следующую за ней по порядку команду, то есть предполагать, что более вероятен отрицательный исход условия». Тогда при выполнении команд условного перехода, в которых высказанное предположение оправдалось, потери времени не будет. В противном случае, заранее извлеченная команда будет удалена из буфера, а вместо нее извлечена правильная, заданная результатом выполнения команда перехода.



Структура правила выполнения условного перехода

Чтобы еще более увеличить выигрыш в быстродействии, на конвейере организуют не две, а большее количество “***рабочих позиций***”, которые разделяют цикл обработки команды на несколько независимых этапов. Традиционно, это позиции:

- извлечение команды (ИК) – чтение из памяти следующей выполняемой машинной команды;

- декодирование команды (ДК) – расшифровка кода операции и спецификаторов операндов;

- вычисление адресов операндов (АО) – вычисление исполнительных адресов операндов-источников;

- извлечение операндов (ИО) – извлечение всех операндов-источников из памяти (операнды, которые ранее помещены в регистр, в выполнении этого этапа не участвуют);

- выполнение команды (ВК) – выполнение операций, заданных кодом операции в команде и, если это задано в команде, сохранение результата в регистрах;

- запись результата (ЗР) – запись результата в память.

Время выполнения перечисленных результатов примерно одинаково. Этот интервал времени называется тактом конвейера

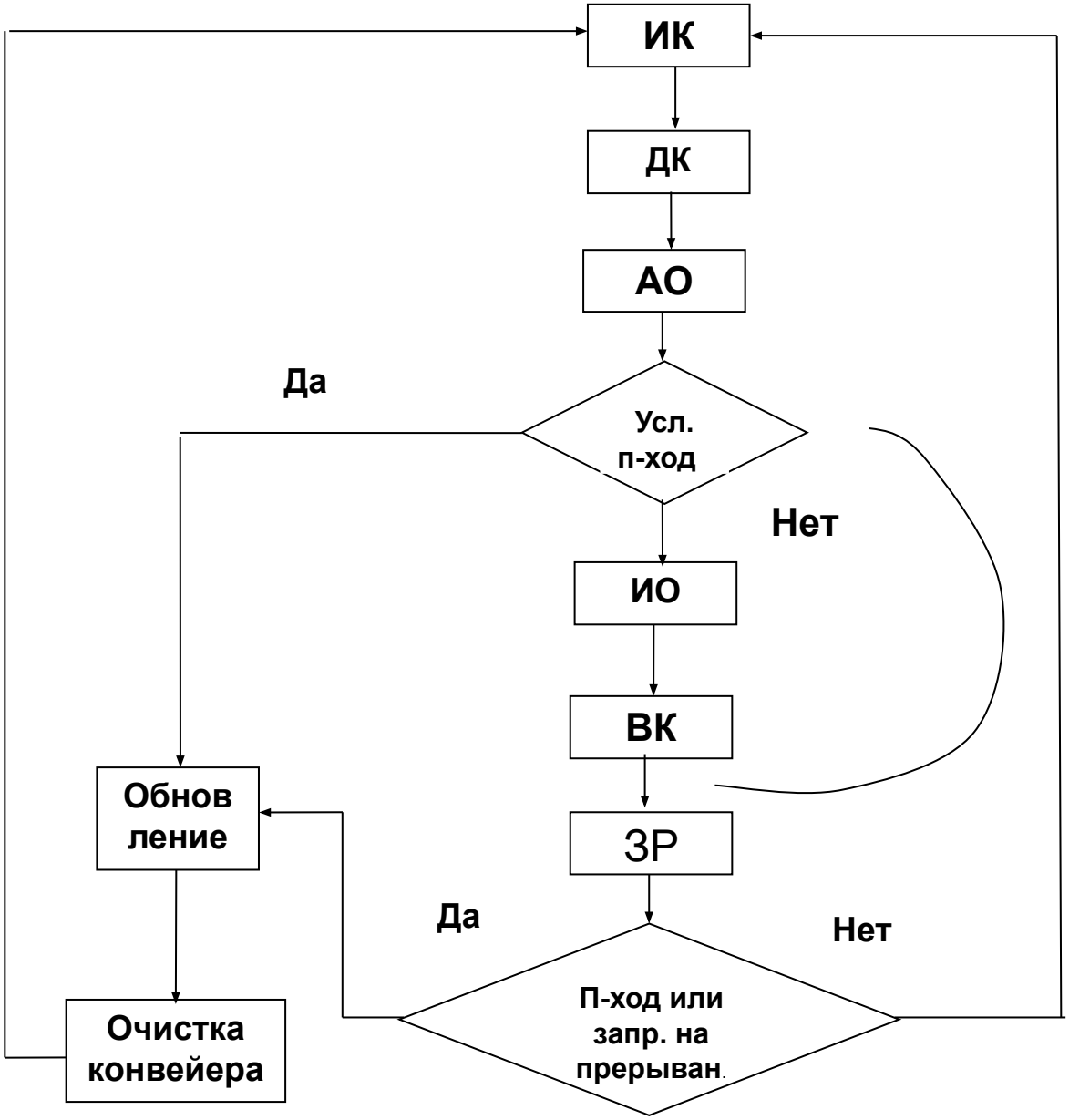
Так, конвейер операций из шести позиций позволяет сократить время выполнения 9 машинных команд с 54 тактов до 14. При построении диаграммы считается, что все этапы выполняются параллельно и что при этом не возникают конфликты при доступе к памяти. Фактически же этапы ***ИК, ИО, ЗР*** предполагают обращение к памяти. С другой стороны, извлекаемая команда может находиться в кэш-памяти или этапы ***ИО*** и ***ЗР*** при выполнении определенных команд могут опускаться. Поэтому, довольно высока вероятность того, что конфликт в борьбе за память не окажет существенного влияния на скорость работы конвейера операций.

Время (такты конвейера) →

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
к1	**ИК**	**ДК**	**АО**	**ИО**	**ВК**	**ЗР**								
К2		**ИК**	**ДК**	**АО**	**ИО**	**ВК**	**ЗР**							
К3			**ИК**	**ДК**	**АО**	**ИО**	**ВК**	**ЗР**						
К4				**ИК**	**ДК**	**АО**	**ИО**	**ВК**	**ЗР**					
К5					**ИК**	**ДК**	**АО**	**ИО**	**ВК**	**ЗР**				
К6						**ИК**	**ДК**	**АО**	**ИО**	**ВК**	**ЗР**			
К7							**ИК**	**ДК**	**АО**	**ИО**	**ВК**	**ЗР**		
К8								**ИК**	**ДК**	**АО**	**ИО**	**ВК**	**ЗР**	
К9									**ИК**	**ДК**	**АО**	**ИО**	**ВК**	**ЗР**

Время (такты конвейера)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
к1	**ИК**	**ДК**	**АО**	**ИО**	**ВК**	**ЗР**								
К2		**ИК**	**ДК**	**АО**	**ИО**	**ВК**	**ЗР**							
К3			**ИК**	**ДК**	**АО**	**ИО**	**ВК**							
К4				**ИК**	**ДК**	**АО**	**ИО**							
К5					**ИК**	**ДК**	**АО**							
К6						**ИК**	**ДК**							
К7							**ИК**							
К8								**ИК**	**ДК**	**АО**	**ИО**	**ВК**	**ЗР**	
К9									**ИК**	**ДК**	**АО**	**ИО**	**ВК**	**ЗР**


ИК
ДК
АО
Да
Усл.
п-ход
Нет
ИО
ВК
Обнов
ление
ЗР
Да
Нет
П-ход или
запр. на
прерыван.
Очистка
конвейера

Для оценки производительности конвейера операций определяется время такта . Это такое время, которое необходимо для продвижения всех команд на одну позицию вперед. Каждая колонка на диаграммах представляет собой один такт конвейера. Длительность такта можно определить по следующим выражениям:

$$\tau = \max \tau_i + d; i = \overline{1, m}.$$

Так как $\tau_i >> d$, то

$$T_k = [k + (n-1)]\tau.$$

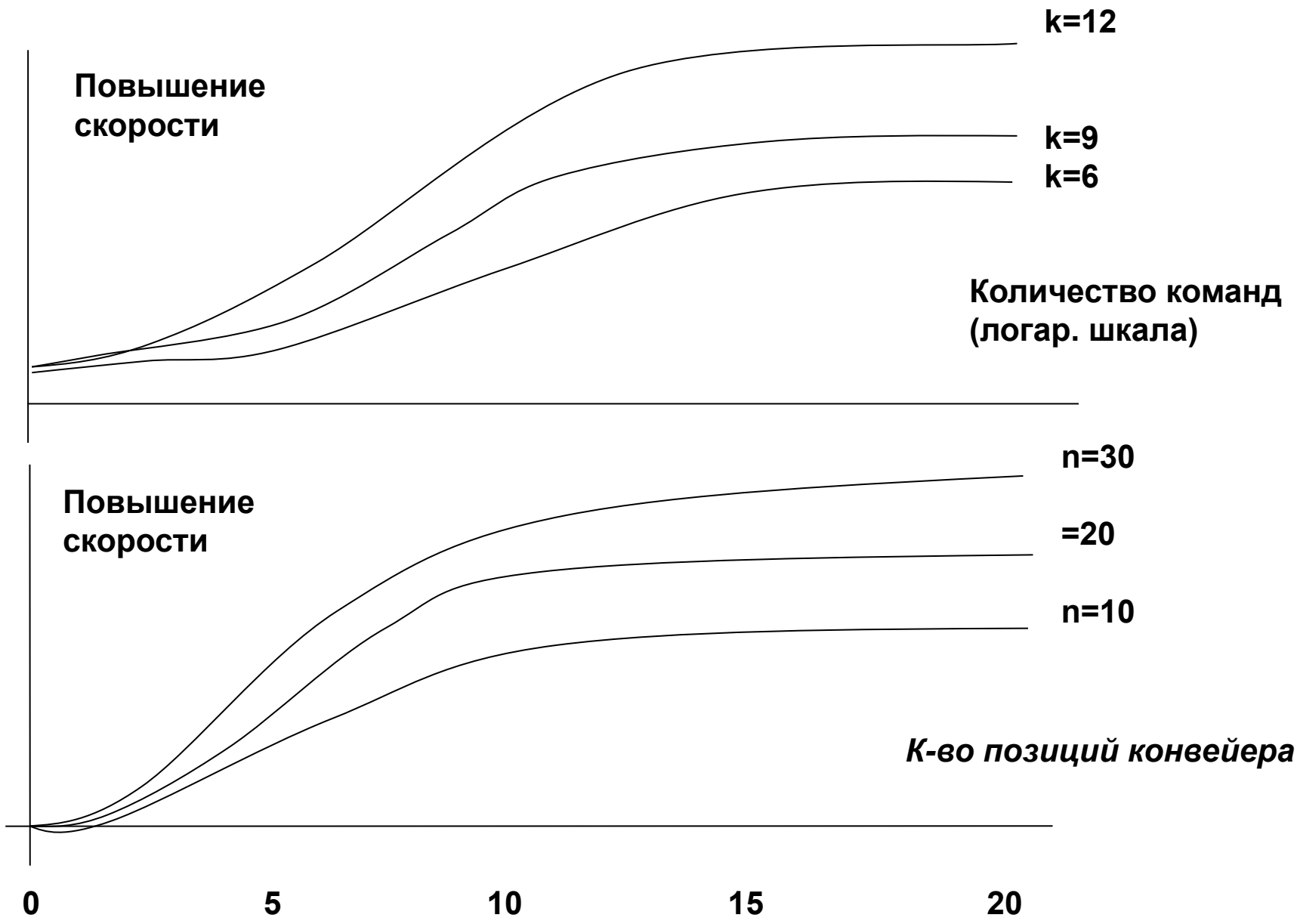
Действительно: T_k=14=**[**k=6+(n=9 -1)**]**, если предположить, что время выполнения одной позиции равно 1.

Коэффициент повышения скорости , благодаря использованию конвейера, оценивается на основании выражения:

$$S_k = \frac{T_1}{T_k} = \frac{nk\tau}{\left[k + (n-1)\right]\tau} = \frac{nk}{k + (n-1)}.$$

Графики, представленные на рисунке а, показывают, как будет изменяться коэффициент повышения скорости при различном количестве команд во фрагменте без команд перехода.

Графики (рис.б) показывают, как изменяется коэффициент повышения скорости в зависимости от количества позиций в конвейере операций. Из графиков видно, что по мере увеличения команд ***n*** повышение скорости стремится к ***k*** в программе без переходов. С другой стороны, при любом количестве команд ***n*** увеличение количества позиций в конвейере операций ведет к увеличению скорости обработки.


k=12
Повышение скорости
k=9
k=6
Количество команд (логар. шкала)
n=30
Повышение скорости
=20
n=10
К-во позиций конвейера
0
5
10
15
20