Департамент компьютерной инженерии

МОДЕЛИРОВАНИЕ РАБОТЫ МНОГОКОНВЕЙЕРНОГО ПРОЦЕССОРА

Методические указания к лабораторной работе

Составитель к.т.н., доц. Е.М. Иванова

Москва 2018

1. ЦЕЛЬ И ПРАКТИЧЕСКОЕ СОДЕРЖАНИЕ МЕТОДИЧЕСКИХ УКАЗАНИЙ

Целью работы является закрепление теоретических знаний по разделам «центральный процессор», «конвейеры».

Краткое содержание

В настоящих указаниях приводится описание особенностей организации работы процессоров при выполнении машинных команд для трёх случаев:

- при отсутствии конвейеров (последовательная архитектура Фон-Неймана),
- при наличии нескольких конвейеров,
- при наличии одного конвейера,

Методические указания и программа содержат достаточное число вариантов заданий. По окончании лабораторной работы студент должен иметь представление о цикле и фазах работы процессора по выполнению команды, микрооперациях и их количестве для различных фаз выполнения команды. Студент должен уметь объяснить полученные графики и таблицы – результат моделирования для своего варианта задания и ответить на вопросы, помещённые в конце данного методического пособия.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В теоретической части данных методических указаний особое внимание уделяется вопросам организации конвейерной обработки информации. Краткий обзор теоретического материала, необходимого в ходе выполнения лабораторной работы, приводится ниже.

2.1. Процедура выполнения команд.

Процессор работает под управлением программы, состоящей из последовательности команд. При выполнении каждой команды процессор выполняет некоторую последовательность действий, называемую циклом выполнения команды. Каждый цикл состоит из нескольких фаз. Стандартные фазы работы ЦП включают в себя [1]:

- Ф1.Выборку команды — передача содержимого счетчика команд в регистр адреса памяти, считывание команды из основной памяти в регистр команды, модификация содержимого счетчика команд для выборки следующей команды.
- Ф2.Декодирование кода операции — для анализа необходимых операций и наличия и местоположения операндов.
- Ф3.Выборку операндов — вычисление адреса и обращение в основную память или к регистру локальной памяти. Операнд считывается и принимается в регистр АЛУ.
- Ф4.Исполнение команды или арифметическая операция — инициализация (кодом операции) цикла работы устройства управления, которое, в свою очередь, управляет работой АЛУ, регистров и схем сопряжения.
- Ф5.Запись результатов — результат выполнения передается в локальную или основную память.

- Ф6.Обработку исключений — сигналов от внутренних блоков, требующих немедленной реакции ЭВМ (если таковые поступили во время выполнения команды или вызваны неверным выполнением самой команды, часто эта фаза не включается в цикл исполнения команды и соответственно в конвейер, т.к. исключение – редкое явление).

В современных процессорах каждая из этих фаз (кроме последней) разбита на более мелкие этапы (2-4шт), что позволяет повысить степень параллелизма и число **исполнительных** потоков.

2.2. Принцип конвейерной обработки

Все фазы цикла исполнения одной машинной команды должны выполняться последовательно (см. рис.1).

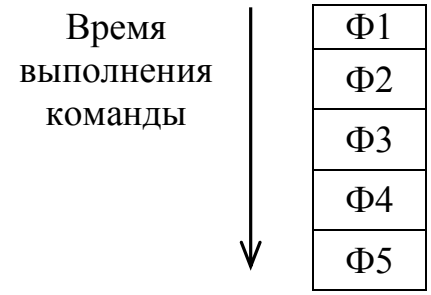


Рис.1. Временная диаграмма загрузки ЦП без выделения блоков (без конвейера)

Рис.2 Общий вид ЦП с одним конвейером команд из 5 ступеней

Каждая фаза команды выполняется определённым блоком ЦП (см. рис.2), который на протяжении цикла исполнения команды работает некоторое время, а остальное время простаивает (см. рис.3).

Загрузка блоков ядра ЦП

	Блок выборки команды	Блок декодирования команд	Блок выборки операндов	Блок исполнения операндов	Блок записи результатов
Время выполнения команды (такты) ↓	Ф1				
		Ф2			
			Ф3		
				Ф4	
					Ф5

Рис.3. Временная диаграмма загрузки блоков ЦП без конвейера

В то время, когда какой-либо блок простаивает, его можно загрузить выполнением фаз другой команды программы. Тогда получается совмещение во времени различных фаз цикла нескольких команд.

Для этого блоки делаются **функционально независимыми** со своими входами и выходами и называются частями - ступенями (уровнями, стадиями) конвейера. А процессор (точнее ядро, отвечающее за исполнение команд программы) называется конвейером.

Совмещение фаз нескольких команд на различных ступенях конвейера приводит к тому, что выполнение следующей команды начинается в нём до окончания предыдущей. Это значительно увеличивает быстродействие процессора. В процессоре может одновременно функционировать несколько конвейеров команд.

Для упрощения модели примем число фаз команды и ступеней конвейера равным 4, соответственно: фаза выборки/декодирования команды, фаза выборки операндов, фаза выполнения операции и фаза записи результатов. На рис.4 показаны временные диаграммы выполнения команд К1…К5 с разбиением на 4 фазы Ф1…Ф4. На рис.5 показана временные диаграммы выполнения пяти команд без конвейерной обработки (слева) и с конвейером (справа).

Здесь можно определить, как время выполнения каждой команды в отдельности (t_1, t_2, …), так и время выполнения всего фрагмента из пяти команд. Как видно, время выполнение К1 и К2 в случае конвейерной обработки не изменилось, но эти интервалы (t_1, t_2) перекрываются. На рис. 5 можно увидеть, что за меньшее время может быть выполнено большее число команд при конвейерной обработке, чем без неё.

Загрузка блоков ЦП

Блок выборки/ декодирования команды	Блок выборки операндов	Блок исполнения операндов	Блок записи результатов
Ф1 К1			
Ф1 К2	Ф2 К2		
Ф1 К3	Ф2 К2	Ф3 К3	
Ф1 К4	Ф2 К3	Ф3 К2	Ф4 К4
Ф1 К5	Ф2 К4	Ф3 К3	Ф4 К2
	Ф2 К5	Ф3 К4	Ф4 К3
		Ф3 К4	Ф4 К5
			Ф4 К5

Время выполнения команды (такты)

t_1

t_2

Рис.4. Временная диаграмма загрузки блоков ЦП с одним конвейером

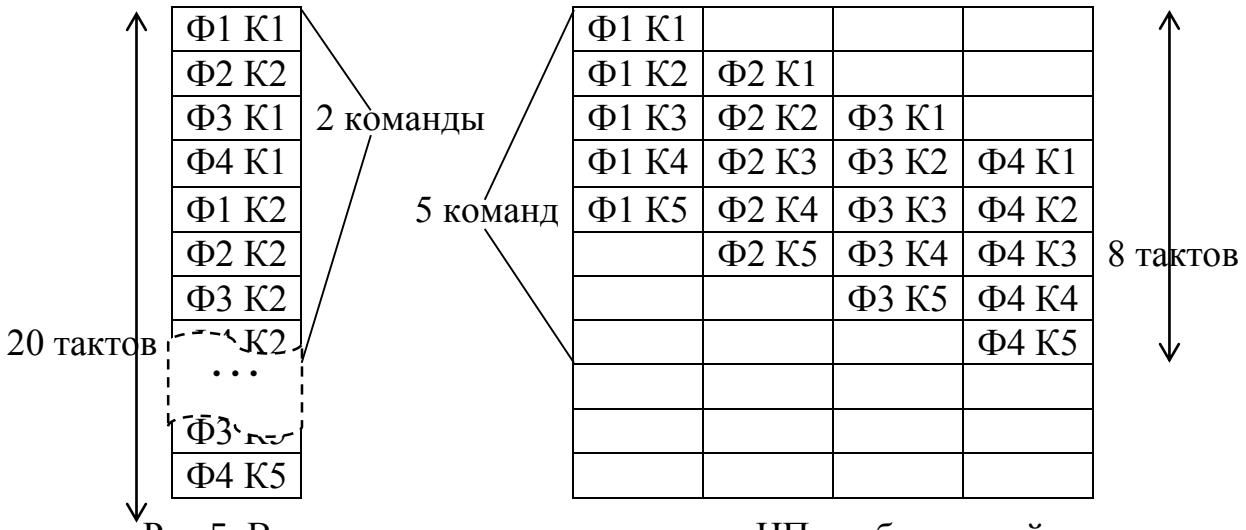


Рис.5. Временная диаграмма загрузки ЦП с и без конвейера

Идея конвейера, давным-давно предложенная Генри Фордом, состоит в том, что производительность цепочки последовательных действий определяется не сложностью этой цепочки, а лишь длительностью самой сложной операции [2]. *Иными словами, совершенно неважно, сколько человек занимаются производством автомобиля и как долго длится его изготовление в целом, - важно то, что если каждый человек в цепочке тратит, скажем, на свою операцию одну минуту, то с конвейера будет сходить один автомобиль в минуту, ни больше и ни меньше; независимо от того, сколько операций нужно совершить с отдельным автомобилем и сколько заняла бы его сборка одним человеком.* Применительно к процессорам принцип конвейера означает, что если мы сумеем разбить выполнение машинной инструкции на несколько этапов, то тактовая частота (а вернее, скорость, с которой процессор забирает данные на исполнение и выдает результаты) будет обратно пропорциональна времени выполнения самого медленного этапа. Если это время удастся сделать достаточно малым (а чем больше этапов на конвейере, тем они короче), то мы сумеем резко повысить тактовую частоту, а значит, и производительность процессора. Именно поэтому число ступеней конвейеров современных процессорных ядер превышает находиться в пределах 15-30.

Совмещенные принципы обработки (конвейер) команд существенно увеличивают производительность (MIPS) процессора, однако эффективность их использования зависит от управления (синхронизации) и числа уровней обработки – ступеней конвейера.

Кроме того, сам программный код существенно влияет на стабильность загрузки ступеней конвейера. Часто в работе ступеней конвейера возникают простои (пузырьки).

Приостановку (простои) работы конвейера вызывает любая команда ветвления, поскольку заранее не известно по какой ветви программа будет выполняться дальше, т.е. не известен адрес следующей *i+1* команды. Он будет определен только по окончании команды *i* передачи управления, и значит, начать заранее выполнение следующей команды мы не можем (см. рис.6). t^* - момент, когда станет известен адрес следующей команды 2, и возможно обращение по этому адресу в ОП для выборки команды.

Простои также возникают при взаимозависимости команд, т.е. использование следующей командой результатов предыдущей команды (см. рис.7). Следующая команда2 не может завершить выборку операндов и начать их обработку, т.к. предыдущая команда1 не записала результат, который и будет операндом для команды2.

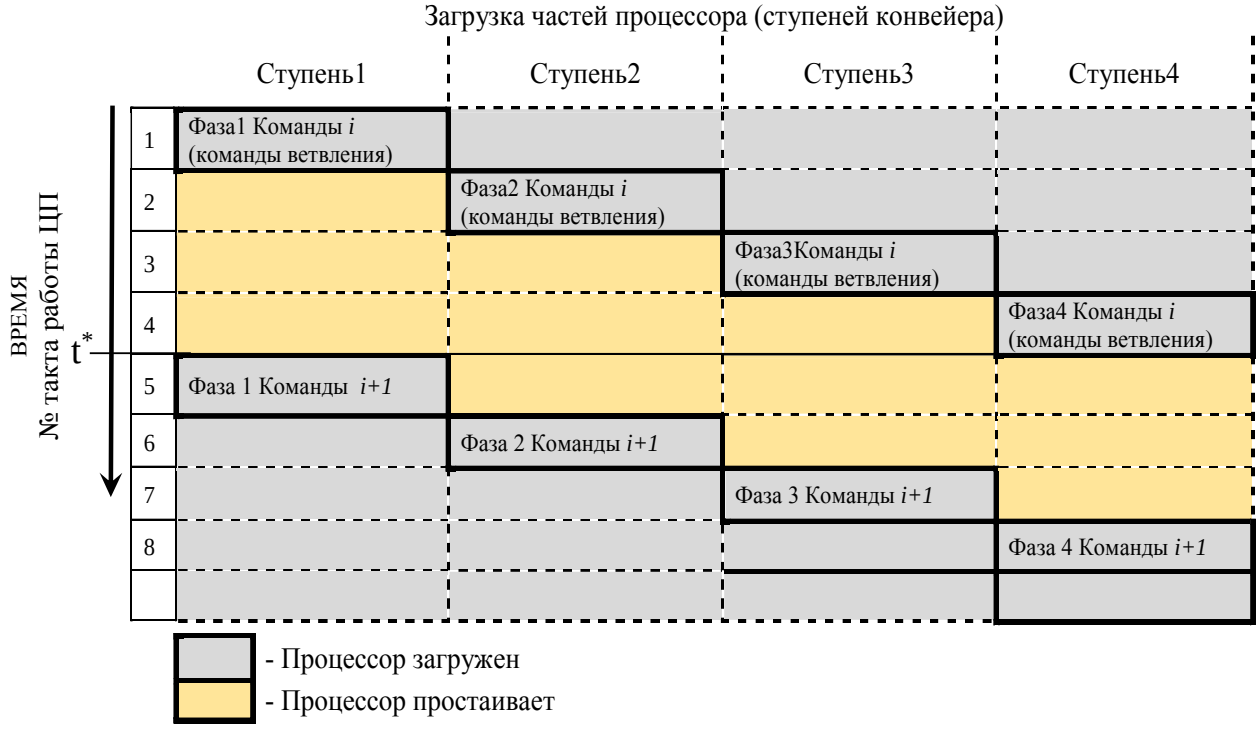


Рис.6. Простой в работе процессора и в программе, вызванный командой ветвления

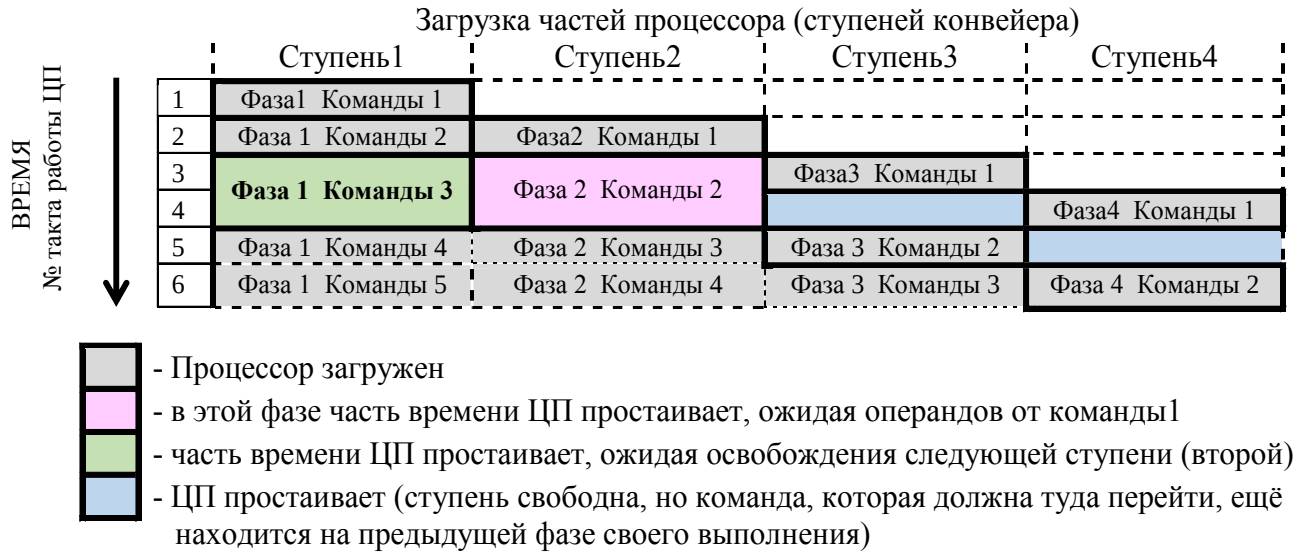


Рис.7. Простой в работе программы, вызванный взаимозависимостью команд

Простои и неравномерность в работе конвейера могут быть также вызваны различной длительностью разных фаз команд (рис.7). Команда3 может уже быть готова перейти к следующей ступени (из фазы1 в фазу2), но эта ступень может оказаться ещё занята предыдущей Командой2. Т.е. в 4 такте простаивает ступень1 (часть ЦП), ступень2, ступень3 и Команда3 (программа). В первом случае причина простоя – «занята следующая ступень конвейера».

И наоборот, ступень конвейера свободна (голубая клетка в 4 и 5 тактах), но команда, которая должна туда перейти (Команда2), ещё находится на предыдущей фазе своего выполнения. Так, например, голубая клетка в 4 такте (третий случай) – простой третьей ступени конвейера (части ЦП), причина которого – «не окончена фаза2 Команды2».

Пустые белые клетки перед началом работы программы (как и после окончания последней команды программы) не считаются простоем, а лишь переходным процессом.

2.3. Система команд

Команда ЭВМ представляет собой двоичный код, определяющий операцию вычислительной машины и данные, участвующие в операции. В явной и неявной форме команда содержит также информацию об адресе, по которому помещается результат операции, и об адресе следующей команды.
Все команды можно разделить на группы по нескольким признакам (см. рис.8).

По назначению все команды можно разбить на следующие группы:

1. **передачи данных.** Сюда следует отнести команды пересылки MOV, загрузки адресов LEA, LDS, LGDT…
2. **обработки данных.** Сюда следует отнести
   команды обработки чисел с фиксированной запятой (ЧФЗ),
   арифметические команды (ADD, SUB, MUL, CMP…)
   логические команды (AND, OR, NEG,…)
   битовые операции над отдельными разрядами операнда (CLI, CLC)
   преобразование (модификация) данных (CBW, CWD…)

команды обработки чисел с плавающей запятой (FCOMP, F2XM1,…)
команды обработки десятичных чисел (DAA, DAS,…)
команды обработки строк (CMPSB, LODS,…)
команды обработки векторов (MASKMOVQ, PADDB,…)

3. **передачи управления** (Jxxx, LOOP, CALL, RET…)
4. **дополнительные команды** (например, команды управления режимами работы процессора)

Рис.8. Классификация команд

Команды разного назначения могут иметь разный алгоритм обработки, разную длительность фаз, разное число и местоположение операндов.

В построенной модели процессора команды обозначены на некотором условном Ассемблере (приближенном к Ассемблеру Intel).

Мнемоника операнд1, операнд2 ; комментарий

Мнемоника означает сокращенное описание выполняемой командой операции

Операнд1 является одновременно и данными для выполнения операции и приёмником результата (если кодом не предполагается иное).

Данными могут быть константы (обозначенные в модели как число или <DATA>), регистры процессора (их обозначение совпадает с обозначением регистров Intel: AX, EDX…), переменные из памяти (обозначенные квадратными скобками [MEM2] или [ESI]), адреса являющиеся в программе метками, но интерпретирующиеся как № команды, т.е. константа (ZERO, STOP…).

В процессе выполнения команд модели формируются служебные данные и адреса, которые «условно» не считаются операндами и результатами, значит на их чтение/запись время при моделировании не учитывается. К таким служебным данным относится состояние регистра указателя команд IP, флагового регистра FLAGS, биты которого (флаги) имеют собственное название и отражают изменения в программе/системе. Однако если одна команда флаг устанавливает, а следующая за ней флаг считывает, то это может стать причиной простоя команды (фазы выборки операндов), хотя все операнды (кроме неявно заданного флага) готовы. Будем использовать те же флаги, что и в процессоре Intel. Назначение разрядов регистра FLAGS приведено ниже в табл. 1, структура показана на рис. 9. Команды передачи данных или передачи

управления флаги не меняют, только команды обработки данных изменяют флаги.

31……22	21	20	19	18	17	16	15	14	13, 12	11	10	9	8	7	6	5	4	3	2	1	0
00…..…0	ID	VIP	VIF	AC	VM	RF	0	NT	IOPL	OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

Рис. 9. Структура регистра FLAGS

Таблица 1. Назначение разрядов регистра флагов FLAGS

Название флага	Назначение флага
CF (Carry Flag)	Флаг переноса устанавливается в 1, если в результате выполнения операции был перенос из старшего разряда
PF (Parity Flag)	Флаг паритета устанавливается в 1, если младшие 8 бит результата операции содержат четное число двоичных единиц.
AF (Auxiliary Flag)	Флаг вспомогательного переноса в старшую тетраду (или заёма) при операциях над упакованными двоично-десятичными числами.
ZF (Zero Flag)	Флаг нуля устанавливается в 1, если результат операции равен нулю.
SF (Sign Flag)	Флаг знака показывает знак результата операции, устанавливаясь в 1 при отрицательном результате.
TF (Trace Flag)	Управляющий флаг трассировки
IF (Interrupt Flag)	Управляющий флаг разрешения прерываний
DF (Direction Flag)	Управляющий флаг направления используется особой группой команд, предназначенных для обработки строк.
OF (Overflow Flag)	Флаг переполнения фиксирует выход результата операции за пределы допустимого для данного процессора диапазона значений.
IOPL (Input/Output Privilege Level)	Двухразрядное поле привилегий в/в указывает на максимальное значение уровня текущего приоритета, при котором команды ввода-вывода выполняются без генерации исключительной ситуации.
NT (Nested Task)	Флаг вложенной задачи.
RF (Restart Flag)	Управляющий флаг рестарта.
VM (Virtual Mode)	Управляющий флаг виртуального режима используется для перевода процессора в режим виртуального процессора 8086.
AC (Alignment Check)	Флаг контроля выравнивания, при установленном флаге, при исполнении программ с уровнем привилегии 3, при обращении к операнду, который не выровнен по соответствующей границе (2, 4, 8 байт) произойдёт исключение #AC.
VIF (Virtual Interrupt Flag)	Флаг разрешения прерывания для многозадачных систем.
VIP (Virtual Interrupt Pending)	Виртуальный запрос прерывания.
ID (ID Flag)	Флаг доступности команды идентификации CPUID.

В настоящей лабораторной работе рассматривается процессор с одним или несколькими **конвейерами команд**. В случае нескольких конвейеров один программный поток разбивается на несколько потоков по правилу: первая команда попадает в первый конвейер, вторая команда – во второй конвейер, … Становится возможным не только совмещение во времени выполнения различных фаз различных команд, но также одинаковых фаз нескольких команд на различных конвейерах.

2.4. Длительность каждой фазы команды

Каждая фаза команды выполняется за определённое число тактов работы процессора. Число тактов, необходимых для выполнения команды (и отдельных ее фаз) зависит от типа операции, от числа её аргументов и от типа аргументов. Для данной лабораторной работы были рассмотрены **условные** параметры. Предположим, что

Фаза1 выборки команды всегда занимает 1 такт.

Фаза2 выборки операндов занимает несколько тактов в зависимости от местоположения операндов.

В системах команд современных ЭВМ многие операции могут выполняться с данными, расположенными в разных ЗУ (см. рис. 10), что значительно упрощает программирование. В этом случае предусматривается возможность использования нескольких способов адресации операндов для одной и той же операции:

- операнд может находиться в самой команде в виде константы,
- в команде может храниться адрес регистра процессора, и операнд выбирается из быстрой внутренней памяти процессора,
- в команде может храниться адрес ячейки ОП, и операнд выбирается из более медленной оперативной памяти ЭВМ.

Рис. 10. Типы памяти для хранения адресуемых данных

Будем для простоты считать, что извлечение операнда из ОП занимает 3 такта, из регистра процессора – 2 такта, для непосредственно заданного операнда (константа, находящаяся в команде) – 1 такт. Если для выполнения команды требуется выбрать 2 операнда, то фаза выборки операндов пропорционально увеличивается. Если операндов нет, тогда выбирается минимальная длительность фазы – 1 такт.

Число тактов, необходимое на **фазу3 исполнения** самой операции, закодированной в команде, указано в табл. 2 [2,4].

Фаза4 записи результатов аналогично фазе выборки операндов занимает 3 или 2 такта в зависимости от местоположения результата (помещение в ОП – 3 такта, занесения в регистр – 2 такта, а константа не может быть изменена). Если результат не записывается, то выбираем самую маленькую длительность фазы – 1 такт.

Таблица 2. Число тактов (микроопераций) для фазы выполнения операции

Команда	Операция	Число тактов

MOV	Переместить - копирование второго операнда на место первого (один операнд, один результат)	2
AND	Логическое умножение двух операндов, результат помещается на место первого операнда (два операнда, один результат)	1
OR	Логическое сложение двух операндов, результат помещается на место первого операнда (два операнда, один результат)	1
XOR	Логическое исключающее «или» двух операндов, результат помещается на место первого операнда (два операнда, один результат)	1
NOT	Инверсия операнда, результат помещается на место операнда	1
NEG	Изменение знака операнда (один операнд, один результат)	1
JMP	Безусловный переход по адресу, указанному в качестве операнда (один операнд – константа=адрес метки, без результата)	2
JZ, JNE, JCXZ, JNA, JNC, … *Jccc*	Условный переход *Jccc <метка>* по адресу, указанному в качестве операнда (метки). При выполнении условия *ccc* (проверки соответствующего флага или комбинации флагов (*например, JZ - при установленном флаге нулевого результата ZF=1*), выполняется переход на метку, иначе выполняется переход к следующей команде, т.е. результатом является служебный адрес перехода (1 такт)	2
SHL/ SHR	Логический сдвиг влево/вправо первого операнда на количество разрядов, заданного вторым операндом (два операнда, один результат)	1
TEST	Логическое умножение двух операндов, результат никуда не помещается, устанавливаются флаги (два операнда, без записи результата)	1
ADD	Арифметическое сложение двух операндов, результат помещается на место первого операнда (два операнда, один результат)	1
ADC	Сложение с переносом (с учётом флага CF переноса), результат помещается на место первого операнда (два операнда+флаг, один результат+флаг)	1
SUB	Вычитание второго операнда из первого, результат помещается на место первого операнда	1
SBB	Вычитание с заёмом (с учётом флага CF переноса), результат помещается на место первого операнда	1
MUL	Умножение числа, расположенного в регистре-аккумуляторе (AL, или AX, или EAX – в зависимости от размера операнда) на операнд	3
DIV	Деление числа, расположенного в регистрах (AX, или DX:AX, или EDX:EAX – в зависимости от размера операнда) на операнд	3
MOVSX /MOVZX	Пересылка со знаковым /с нулевым расширением второго операнда на место первого	2
CMP	Сравнение двух операндов путём вычитания второго из первого и установка флагов, результат операции никуда не заносится	1
INC/DEC	Увеличение /уменьшение на 1 содержимого операнда	1
PUSH	Уменьшение значения ESP на 4 и занесение в стек операнда	1
POP	Извлечение из стека операнда, увеличение ESP на 4	1
PUSHFD	Уменьшение значения ESP на 4 и сохранение флагового регистра FLAGS в стеке по адресу SS:ESP.	1
POPFD	Извлечение из стека флагового регистра EFLAGS, увеличение ESP на 4.	1
LOOP	Команда организации цикла, уменьшить содержимое CX на 1, сравнить с нулём, если CX>0 перейти по адресу, указанному в качестве операнда (по метке), иначе к следующей команде	2
CLD/STD	Сбросить/установить флаг направления просмотра строк DF=0/1	1
CMPSB	Сравнение байтов, расположенных в памяти по логическим адресам	2

	DS:ESI, ES:EDI, по результату сравнения установить флаги	
CALL	Вызов процедуры (подпрограммы), занести в стек адрес возврата EIP	2
RET	Возврат из процедуры, восстановление из стека в EIP адреса возврата	3
NOP	Пустая операция, увеличивает EIP	1
LEA	Загрузка исполнительного адреса области ОП для второго операнда в первый операнд	1

3. ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

В работе требуется для заданной последовательности команд (фрагмента кода на Ассемблере) промоделировать работу процессора в трёх случаях:

1. без конвейера (вручную, заполнив таблицу 3),
2. при наличии одного конвейера (с помощью программы),
3. при наличии нескольких конвейеров (с помощью программы).

Для случая 1 самостоятельно просчитать время выполнения каждой команды, которое будет складываться из длительности четырёх фаз (см. раздел 2.4), и всего фрагмента программы. Оформить в виде таблицы3. Обратите внимание на то, что в программе может присутствовать цикл, тогда команды из тела цикла надо учесть несколько раз. Также следует понимать, что при ветвлении не все команды программы выполняются, и значит учитывать в общем времени команды из не выбранной ветки не нужно.

Результатами моделирования одноконвейерного и многоконвейерного процессоров будут временные диаграммы загрузки одного или нескольких конвейеров процессора.

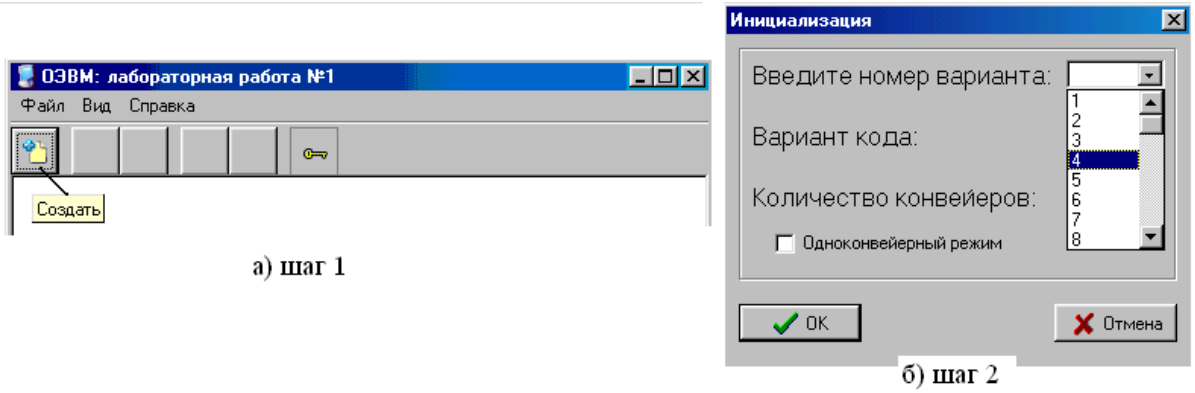
Таблица 3. Время выполнения фрагмента без конвейера

команда	Классификация по назначению	Местоположение операндов	Местоположение результатов	длительность фазы				длительность команды
				Ф1	Ф2	Ф3	Ф4	
…	**…**	**Оп1… Оп2… Оп3…**	**Рез1… Рез2…**	**…**	**…**	**…**	**…**	**…**
Общее время выполнения фрагмента кода без конвейера								

Кроме того, по ходу моделирования отражается положение команд в конвейере, их расчётное и фактическое время пребывания на каждой ступени. Зная время, требуемое на выполнение фазы (табл.3), можно проследить загрузку блоков ЦП и возникающие простои.

4. ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В качестве задания каждый студент имеет № варианта, который выдаётся преподавателем лично (совпадает с № по списку группы).



а) шаг 1 б) шаг 2

Рис.11. Инициализация моделирования

Инициализация. После запуска программы на исполнение на экране появится окно моделирующей программы, выберите закладку - «Создать», (см. рис.11,а) на ней – номер заданного варианта и режим моделирования: при моделировании в одноконвейерном режиме поставьте пометку в соответствующем поле, нажмите на кнопку «ОК» (см. рис.11,б).

В окне моделирования появится заданный для моделирования фрагмент кода на Ассемблере (см.рис.12). В первых позициях фрагмента (первый столбец) могут располагаться метки, далее идёт столбец, содержащий условное обозначение команды (мнемонику), в третьем столбце располагаются операнды, отделённые друг от друга запятой, затем следуют комментарии, начинающиеся с символа «;». Изучите фрагмент программы. Разберитесь в выполняемых действиях. Классифицируйте каждую команду по назначению и количеству адресов в адресной части команды: определите число и местоположение операндов и результата.

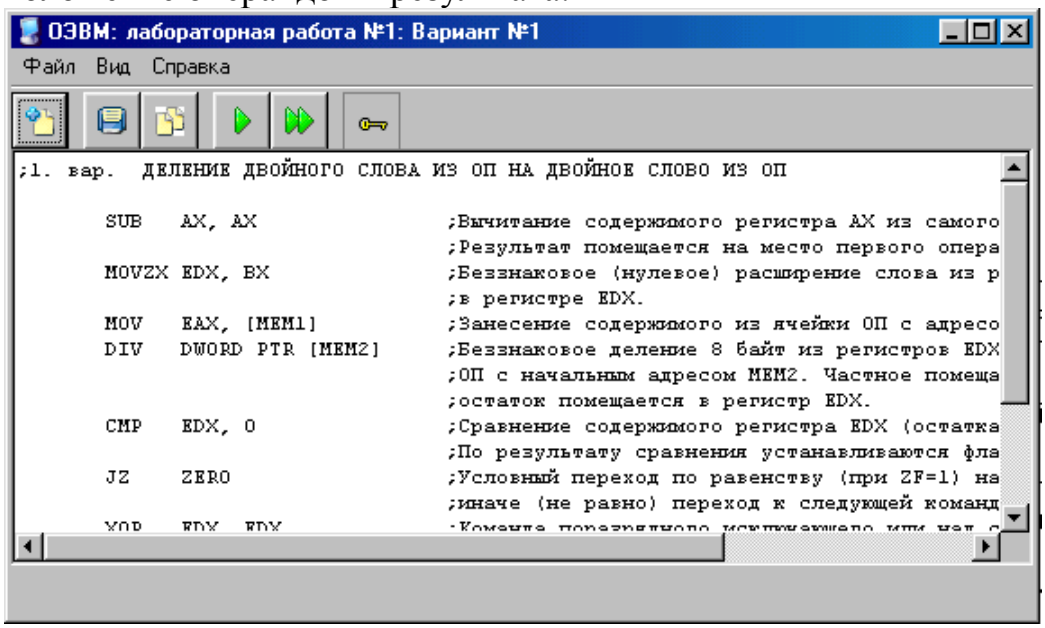


Рис.12. Инициализация моделирования, шаг 3

Для этого фрагмента требуется проследить выполнение заданной последовательности команд в многоконвейерном процессоре и сравнить с выполнением фрагмента в одноконвейерном процессоре. Последовательность выполнения команд без использования конвейера проанализировать самостоятельно и сравнить с двумя временными диаграммами.

Начало моделирования. Кнопки, расположенные в меню окна моделирующей программы, имеют общеизвестный смысл:

 - сохранить,  - копировать,

 - следующий такт в режиме пошагового моделирования,

 - выполнить все инструкции (всю программу целиком).

Далее, нажав на кнопку «Следующий такт», можно начинать выполнять моделирование работы многоконвейерного (одноконвейерного) процессора. Каждое последующее нажатие на кнопку «Следующий такт» отразит исполнение в процессоре следующей микрооперации.

Просмотреть изменения, происходящие в конвейере, можно, использовав меню моделирующей программы, опцию «Вид» (рис.13). Она даёт возможность открыть дополнительные окна:

- временная диаграмма выполнения команд,
- положение команд в конвейерах

и внести некоторую коррекцию в изображение по желанию пользователя.

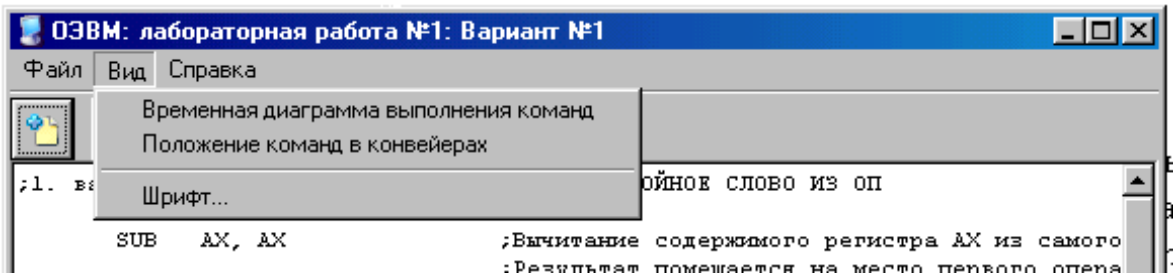


Рис.13. Инициализация моделирования, шаг 4

После инициализации в конвейере микропроцессора не находится ни одной микрооперации о чем говорят пустые ячейки таблицы в окне «Положение команд в конвейерах».

Строки таблицы «Положение команд в конвейерах» соответствуют четырём ступеням каждого моделируемого конвейера или четырём фазам выполнения команды в процессоре (см. рис.14):

- ступень1 - «фаза выборки команды»,
- ступень2 - «фаза выборки операндов»,
- ступень3 - «фаза исполнения операции»,
- ступень4 - «фаза записи результатов».

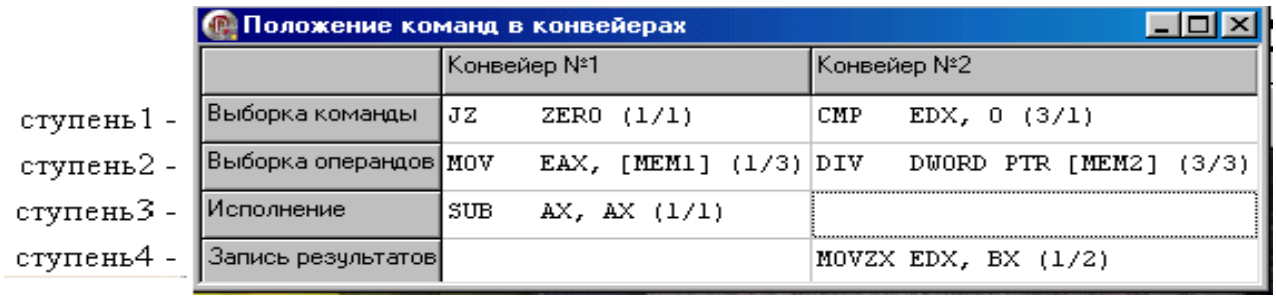
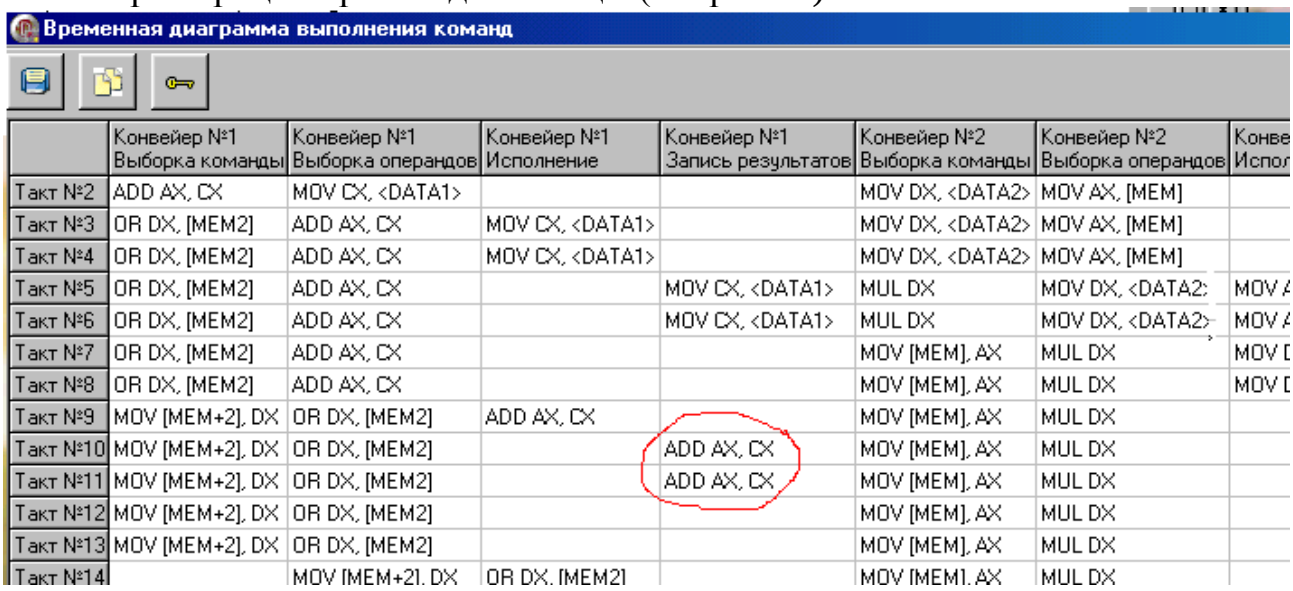


Рис. 14. Окно «Положение команд в конвейерах»

Столбцы таблицы соответствуют номеру используемого конвейера (в случае многоконвейерного моделирования). По мере выполнения команд в ячейках таблицы будут появляться мнемоники той команды, которая в текущий момент времени (на данном такте работы процессора, номер которого отражён в окне «временная диаграмма выполнения команд») занимает указанную ступень указанного конвейера. Кроме того, после каждой команды в скобках указаны два числа (например, «(1/3)»), что означает, что в данной фазе команда должна пробыть 3 такта, а пробыла уже 1 такт (см. рассчитанную табл.3).

В окне «Временная диаграмма загрузки конвейеров» представлено графическое отображение последовательности исполнения команд на конвейерах процессора в виде таблицы (см. рис.15).



	Конвейер №1 Выборка команды	Конвейер №1 Выборка операндов	Конвейер №1 Исполнение	Конвейер №1 Запись результатов	Конвейер №2 Выборка команды	Конвейер №2 Выборка операндов	Конве Испол
Такт №2	ADD AX, CX	MOV CX, <DATA1>			MOV DX, <DATA2>	MOV AX, [MEM]	
Такт №3	OR DX, [MEM2]	ADD AX, CX	MOV CX, <DATA1>		MOV DX, <DATA2>	MOV AX, [MEM]	
Такт №4	OR DX, [MEM2]	ADD AX, CX	MOV CX, <DATA1>		MOV DX, <DATA2>	MOV AX, [MEM]	
Такт №5	OR DX, [MEM2]	ADD AX, CX		MOV CX, <DATA1>	MUL DX	MOV DX, <DATA2>	MOV A
Такт №6	OR DX, [MEM2]	ADD AX, CX		MOV CX, <DATA1>	MUL DX	MOV DX, <DATA2>	MOV A
Такт №7	OR DX, [MEM2]	ADD AX, CX			MOV [MEM], AX	MUL DX	MOV D
Такт №8	OR DX, [MEM2]	ADD AX, CX			MOV [MEM], AX	MUL DX	MOV D
Такт №9	MOV [MEM+2], DX	OR DX, [MEM2]	ADD AX, CX		MOV [MEM], AX	MUL DX	
Такт №10	MOV [MEM+2], DX	OR DX, [MEM2]		ADD AX, CX	MOV [MEM], AX	MUL DX	
Такт №11	MOV [MEM+2], DX	OR DX, [MEM2]		ADD AX, CX	MOV [MEM], AX	MUL DX	
Такт №12	MOV [MEM+2], DX	OR DX, [MEM2]			MOV [MEM], AX	MUL DX	
Такт №13	MOV [MEM+2], DX	OR DX, [MEM2]			MOV [MEM], AX	MUL DX	
Такт №14		MOV [MEM+2], DX	OR DX, [MEM2]		MOV [MEM], AX	MUL DX	

Рис.15. Временная диаграмма загрузки конвейеров

По вертикальной оси (сверху вниз) отражается номер такта работы процессора. По горизонтальной оси (слева направо) показаны ступени используемых конвейеров. На пересечении можно увидеть мнемонику команды, которая в указанном такте исполняется на указанной ступени указанного конвейера. Например, обведенный на рис.15 фрагмент означает, что команда ADD с двумя операндами AX, CX пробыла в фазе записи результата 2 такта, поскольку результат пишется на место первого операнда AX (регистр).

Поскольку получающиеся временные диаграммы довольно велики и целиком не помещаются на экране монитора, то для перемещения просмотра

нужной части диаграммы воспользуйтесь линейками прокрутки: «вверх», «вниз», «вправо», «влево».

Нажимая кнопку «Следующий такт» и анализируя информацию в различных окнах, до тех пор, пока не перестанет увеличиваться число строк в таблице, можно проследить весь ход моделирования конвейерной обработки нескольких команд для заданного фрагмента кода. Нажав кнопку «выполнить все инструкции», можно получить всю временную диаграмму полностью. Следует обратить внимание на длительность пребывания (необходимую и фактическую) каждой команды на каждой ступени конвейера, а также на реальную загрузку процессора и простои в ожидании готовности к выполнению очередной фазы команды и причины этих простоев.

Простои в работе конвейера можно выявить следующим образом:

- пустые клеточки на временной диаграмме выполнения команд (*за исключением клеток перед первой командой и после последней команды – начало и конец работы конвейера – переходный процесс не отражает загрузку конвейера во время выполнения программы)*

Причины:

1) Разные фазы разных команд исполняются за разное время и данные с предыдущей ступени ещё не готовы для перехода команды на следующую. Как правило, это происходит после наиболее длительной фазы команды – выборки операндов (см. рис.7, голубые клетки).

2) Не окончена команда передачи управления (J*xxx*, LOOP, CALL, RET…). После таких команд возникают большие простои (набор пустых клеток) в несколько тактов уже начиная с фазы выборки команды и распространяющиеся на все ступени конвейера. Результатом таких команд является адрес следующей команды, и до окончания команды передачи управления процессору неизвестно, какую следующую команду надо выбирать (см. рис.6, желтые клетки).

- команда находится в какой-либо фазе дольше, чем положено *(сравнить длительность фаз с диаграммы и из табл.3)*.

Причины:

3) Следующая ступень конвейера, куда должна перейти команда2, занята исполнением предыдущей команды1. Такое обычно происходит перед самой длительной фазой команды – выборки операндов (см. рис.7, зеленые клетки).

4) Операнды (регистры, в том числе и флаговый, ячейки памяти, стек), требуемые для исполнения текущей команды, используются другими предыдущими командами, возможно, в другом конвейере (см. рис.7, розовые клетки). Очередность команд проверить по исходному тексту фрагмента кода.

5. СОДЕРЖАНИЕ ОТЧЁТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

В отчёт по лабораторной работе следует включить:

1. Вариант задания с текстом фрагмента кода на Ассемблере с комментариями,
2. Заполненная табл.3 согласно разделу 2.4,

3. Временные диаграммы загрузки одного и нескольких конвейеров процессора,
4. Анализ результатов моделирования (отметить на диаграммах (например, разным цветом) различные виды простоя конвейера с объяснениями их причин: 4 причины – 4 цвета).

Пример оформления отчёта по ЛР приведен в приложении.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

Для защиты работы вам нужно будет ответить на несколько вопросов

1. Перечислить фазы цикла выполнения команды в процессоре.
2. Что такое микрооперация?
3. Зачем разбивать цикл выполнения команды на отдельные фазы?
4. Что такое конвейерная обработка информации?
5. Что такое конвейер команд?
6. Что такое арифметический конвейер?
7. В чём заключается отличие арифметического конвейера от конвейера команд?
8. Для чего понадобилось организовывать конвейеры?
9. Что такое ступень конвейера?
10. От чего зависит производительность процессора с конвейером команд?
11. Почему для одинаковых команд требуется различное число тактов на выборку операндов из регистра, памяти и при непосредственной адресации?
12. Что такое простои конвейера?
13. Назовите причины простоев.
14. Подумайте, как можно было бы устранить простои разных типов?
15. Что такое команда ЭВМ?
16. Как команды различаются по назначению?
17. Что такое операнды?
18. Перечислите (с примерами из заданного фрагмента кода на Ассемблере) режимы адресации операндов?
19. Обязательно ли все операнды указываются в адресной части команды, пояснить на примере?
20. Как вы думаете, почему время выборки операндов размером 1, 2 или 4 байта одинаковое?

7. СПИСОК ЛИТЕРАТУРЫ

1. Таненбаум, Э. Архитектура компьютера. СПб. Питер, 2014. - 811 с.

2. Сергей Озеров. Архитектура процессоров. Конвейер, суперскалярные и Out-of-Order-процессоры. Компьютерра Online, 26 октября 2005 года, http://old.computerra.ru/system/235537/

3. Материалы со страницы изучаемой дисциплины на сайте LMS

ПРИЛОЖЕНИЕ
Пример оформления отчёта по ЛР

Вариант XX

Фрагмент кода программы

```
        MOV EAX,[MEM] ;пересылка операнда из ОП в регистр EAX
        ADD EAX,6     ;сложение двух операндов из регистра EAX и константы 6Ю, результат в EAX
        JZ  MMM       ;передача управления на метку MMM при условии нулевого результата на
                      ;предыдущем шаге, иначе переход к следующей команде
        OR [MEM],0001H  ;логическое сложение (ИЛИ) операнда из ОП с 16-й константой 0001, результат в ОП
MMM:    MOV [MEM],<DATA> ;пересылка непосредственно заданной константы <DATA> в ОП
```

Таблица 2. Время выполнения фрагмента без конвейера

команда	Классификация по назначению	Количество и местоположение операндов	Местоположение результата	длительность фазы				Длительность команды
				Ф1	Ф2	Ф3	Ф4	
MOV EAX,[MEM]	Передачи данных	оп1 в ОП	Рез1 в регистре	1	3	2	2	8
ADD EAX,6 **Зависимые по данным команды**	Обработки данных	Оп1 в регистре, Оп2 (константа) в команде Оп3 флаги (чтение не требуется)	Рез1 в регистре EAX, Рез2 – флаги (запись не требуется	1	1+2=3	1	2	7
JZ MMM	Передачи управления	оп1 (константа) в команде Оп2 IP (чтение не требуется)	Рез1 IP (запись не требуется)	1	1	2	~~0~~1	5
OR [MEM],0001H	Обработки данных	Оп1 в ОП, Оп2 (константа) в команде Оп3 флаги (чтение не	Рез1 в ОП Рез2 – флаги (запись не требуется	1	3+1=4	1	3	9

		требуется)						
MOV [MEM],<DATA>	Передачи данных	Оп1 (константа) в команде	Рез1 в ОП	1	1	2	3	7
Общее время выполнения фрагмента кода без конвейера								36

2)временная диаграмма загрузки одного конвейера

№ такта	Ступень 1. Блок выборки команды	Ступень 2. Блок выборки операндов	Ступень 3. Блок исполнения операции	Ступень 4. Блок записи результатов
1.	MOV EAX,[MEM]			
2.	ADD EAX,6	MOV EAX,[MEM]		
3.	ADD EAX,6	MOV EAX,[MEM]		
4.	ADD EAX,6	MOV EAX,[MEM]		
5.	JZ MMM	ADD EAX,6	MOV EAX,[MEM]	
6.	JZ MMM	ADD EAX,6	MOV EAX,[MEM]	
7.	JZ MMM	ADD EAX,6		MOV EAX,[MEM]
8.		ADD EAX,6		MOV EAX,[MEM]
9.		JZ MMM	ADD EAX,6	
10.		JZ MMM		ADD EAX,6
11.		JZ MMM		ADD EAX,6
12.			JZ MMM	
13.			JZ MMM	
14.				JZ MMM
15.	OR [MEM],0001H			
16.	MOV [MEM],<DATA>	OR [MEM],0001H		
17.	MOV [MEM],<DATA>	OR [MEM],0001H		
18.	MOV [MEM],<DATA>	OR [MEM],0001H		
19.	MOV [MEM],<DATA>	OR [MEM],0001H		
20.		MOV [MEM],<DATA>	OR [MEM],0001H	
21.			MOV [MEM],<DATA>	OR [MEM],0001H
22.			MOV [MEM],<DATA>	OR [MEM],0001H

23.			MOV [MEM],<DATA>	OR [MEM],0001H
24.				MOV [MEM],<DATA>
25.				MOV [MEM],<DATA>
26.				MOV [MEM],<DATA>

	Простои вызванные командой передачи управления, ЦП не знает адрес передачи управления (адрес следующей команды)
	Простои вызванные различной длительностью фаз соседних по конвейеру команд (ADD EAX,6 и MOV EAX,[MEM]) – следующая ступень (блок выборки операндов), куда должна перейти команда ADD EAX,6 занята командой MOV EAX,[MEM]
	Простои вызванные различной длительностью фаз соседних по конвейеру команд (ADD EAX,6 и MOV EAX,[MEM]) хотя фаза исполнения команды MOV EAX,[MEM] уже завершена и блок исполнения операции свободен, команда ADD EAX,6 ещё не окончила выборку операндов и не готова перейти в освободившийся блок, который и будет простаивать
	Простой, вызванный зависимостью команд по данным: для команды ADD EAX,6 нужно выбрать два операнда. Константа 6 из команды может быть выбрана сразу (5 такт – 2 ступень), а вот операнд из регистра EAX может быть использован только после окончания записи в него результата предыдущей команды MOV EAX,[MEM], т.е. после 8 такта. Т.к. результат предыдущей команды уже находится буфере исполнительных устройств, то повторная выборка нге происходит, а на 9 такте сразу начинается исполнение операции сложения.

3)временная диаграмма загрузки двух конвейеров

№	Ступень 1 Конвейера1	Ступень 2. Конвейера1	Ступень 3. Конвейера1	Ступень 4. Конвейера1	Ступень 1 Конвейера2	Ступень 2. Конвейера2	Ступень 3. Конвейера2	Ступень 4. Конвейера2	№
1.	MOV EAX,[MEM]				ADD EAX,6				1.
2.	JZ MMM	MOV EAX,[MEM]				ADD EAX,6			2.
3.	JZ MMM	MOV EAX,[MEM]				ADD EAX,6			3.
4.	JZ MMM	MOV EAX,[MEM]				ADD EAX,6			4.
5.		JZ MMM	MOV EAX,[MEM]			ADD EAX,6			5.
6.		JZ MMM	MOV EAX,[MEM]			ADD EAX,6			6.
7.		JZ MMM		MOV EAX,[MEM]		ADD EAX,6			7.
8.		JZ MMM		MOV EAX,[MEM]		ADD EAX,6			8.
9.		JZ MMM					ADD EAX,6		9.
10.		JZ MMM						ADD EAX,6	10.
11.		JZ MMM						ADD EAX,6	11.
12.			JZ MMM						12.
13.			JZ MMM						13.

14.				JZ MMM					14.
15.	MOV [MEM],<DATA>				OR [MEM],0001H				15.
16.		MOV [MEM],<DATA>				OR [MEM],0001H			16.
17.			MOV [MEM],<DATA>			OR [MEM],0001H			17.
18.			MOV [MEM],<DATA>			OR [MEM],0001H			18.
19.				MOV [MEM],<DATA>		OR [MEM],0001H			19.
20.				MOV [MEM],<DATA>			OR [MEM],0001H		20.
21.								OR [MEM],0001H	21.
22.								OR [MEM],0001H	22.
23.								OR [MEM],0001H	23.

	Простои вызванные командой передачи управления, ЦП не знает адрес передачи управления (адрес следующей команды) длительность такого простоя увеличивается и расширяется на оба конвейера, т.к. команды после команды передачи управления распределились по обоим конвейерам
	Простои вызванные различной длительностью фаз соседних по конвейеру команд. Таких простоев стало меньше, т.к если заняты ступени текущего конвейера, то можно задействовать второй конвейер
	Простои вызванные различной длительностью фаз соседних по конвейеру команд (команда не готова перейти в освободившийся блок). Из-за взаимозависимости команд на разных конвейерах такие простои удлинились
	Простой, вызванный зависимостью команд по данным: из-за взаимозависимости команд на разных конвейерах такие простои удлинились. Например, для команды ADD EAX,6 выборка/использование операнда из регистра EAX может быть начато только на 9 такте на ступени2 конвейера2 (после завершения команды MOV EAX,[MEM] на такте 8 в конвейере1). И хотя выборка команды на параллельном конвейере начата раньше, чем в случае одноконвейерного процессора, но завершиться она в то же время. Простой увеличился на 3 такта.

Вывод: *уж это напишите сами, сравнив последовательную обработку с одноконвейерной и один конвейер с несколькими*