АППАРАТНЫЕ СРЕДСТВА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНКИКИ

Галимов Р.Р.,2013

Распределение учебной нагрузки

- Лекции -9 занятий
- Практические занятия -9 занятий
- Лабораторные работы - 18 занятий
- Итоговый контроль -экзамен, РГЗ

ЛИТЕРАТУРА

1. Калабеков Б.А. Цифровые устройства и микропроцессорные системы / Б.А. Калабеков - М.: Горячая линия – Телеком, 2002.-336с.
2. Новиков Ю.В. Основы микропроцессорной техники / Ю.В. Новиков, П.К. Скоробогатов - М.: ИНТУИТ.РУ. «Интернет-Университет Информационных технологий», 2006.-440с.
3. Пухальский Г.И. Проектирование микропроцессорных устройств: Учебное пособие для вузов / Г.И. Пухальский - СПб.: Политехника, 2001-544с.
4. Микропроцессорный комплект К1810: структура, программирование, применение [Текст] : справ. книга / под ред. Ю. М. Казаринова . - М. : Высш. шк., 1990. - 269 с.

ЖУРНАЛЫ

- CHIP NEW;
- BYTE;
- Мир компьютерной автоматизации;
- Современная электроника;
- Железо
- Хакер

САЙТЫ:

1. www.ixbt.com

Лекция №1. Основные понятия. Микропроцессор (МП), микропроцессорная система

Рисунок 1.1 – Методы обработки информации

Достоинства цифровой обработки информации:

- возможность обеспечения любой точности обработки;
- высокая помехозащищенность;
- высокая стабильность характеристик обработки;
- возможность выполнения таких видов обработки, которые аналоговыми методами либо трудновыполнимы, либо вовсе невыполнимы.

Недостаток: относительно низкое быстродействие.

Общие принципы построения современных ЭВМ

Основные принципы построения ЭВМ были сформулированы американским учёным Джоном фон Нейманом в 40-х годах 20 века:

1. Любую ЭВМ образуют три основные компоненты: процессор, память и устройства ввода-вывода (УВВ).
2. Информация, с которой работает ЭВМ делится на два типа:
   - набор команд по обработке (программы);
   - данные подлежащие обработке.
3. И команды, и данные вводятся в память (ОЗУ) – **принцип хранимой программы**.
4. Руководит обработкой процессор, устройство управления (УУ) которого выбирает команды из ОЗУ и организует их выполнение, а арифметико-логическое устройство (АЛУ) проводит арифметические и логические операции над данными.
5. С процессором и ОЗУ связаны устройства ввода-вывода (УВВ).

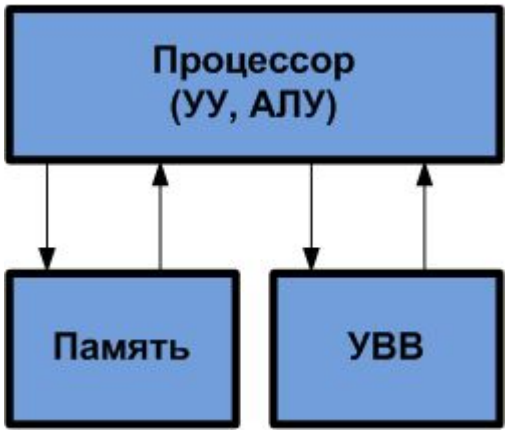


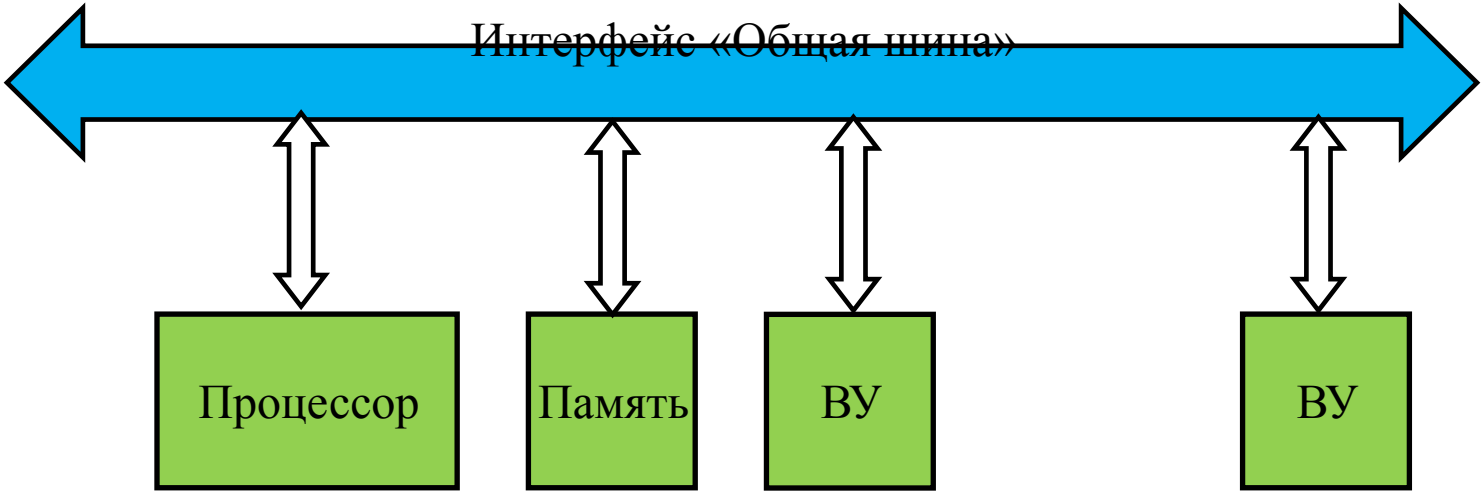
Рисунок 1.2 – Общая структура ЭВМ

Микропроцессор. Микропроцессорная система

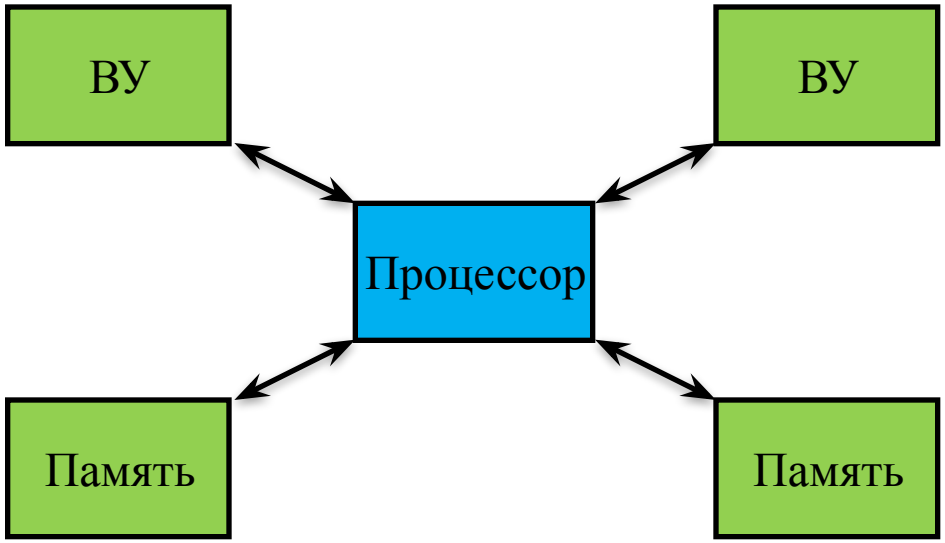
Микропроцессором называется программно-управляемое устройство для обработки цифровой информации и управления процессом обработки, реализованное в виде большой (БИС) или сверхбольшой (СБИС) интегральной микросхемы.

Микропроцессорная система (МПС) представляет собой функционально законченное изделие, состоящее из одного или нескольких устройств, главным образом микропроцессорных:

- микропроцессора и/или микроконтроллера;
- память;
- устройства ввода/вывода;
- вспомогательные схемы (тактовый генератор, контроллеры прерываний и ПДП, шинные формирователи, регистры-защелки и др.).



Структура МПС с интерфейсом "Общая шина"



Структура МПС с радиальными линиями связи

Характеристики структур современных ЭВМ

- **модульность построения;**
- **магистральность;**
- **иерархия управления.**

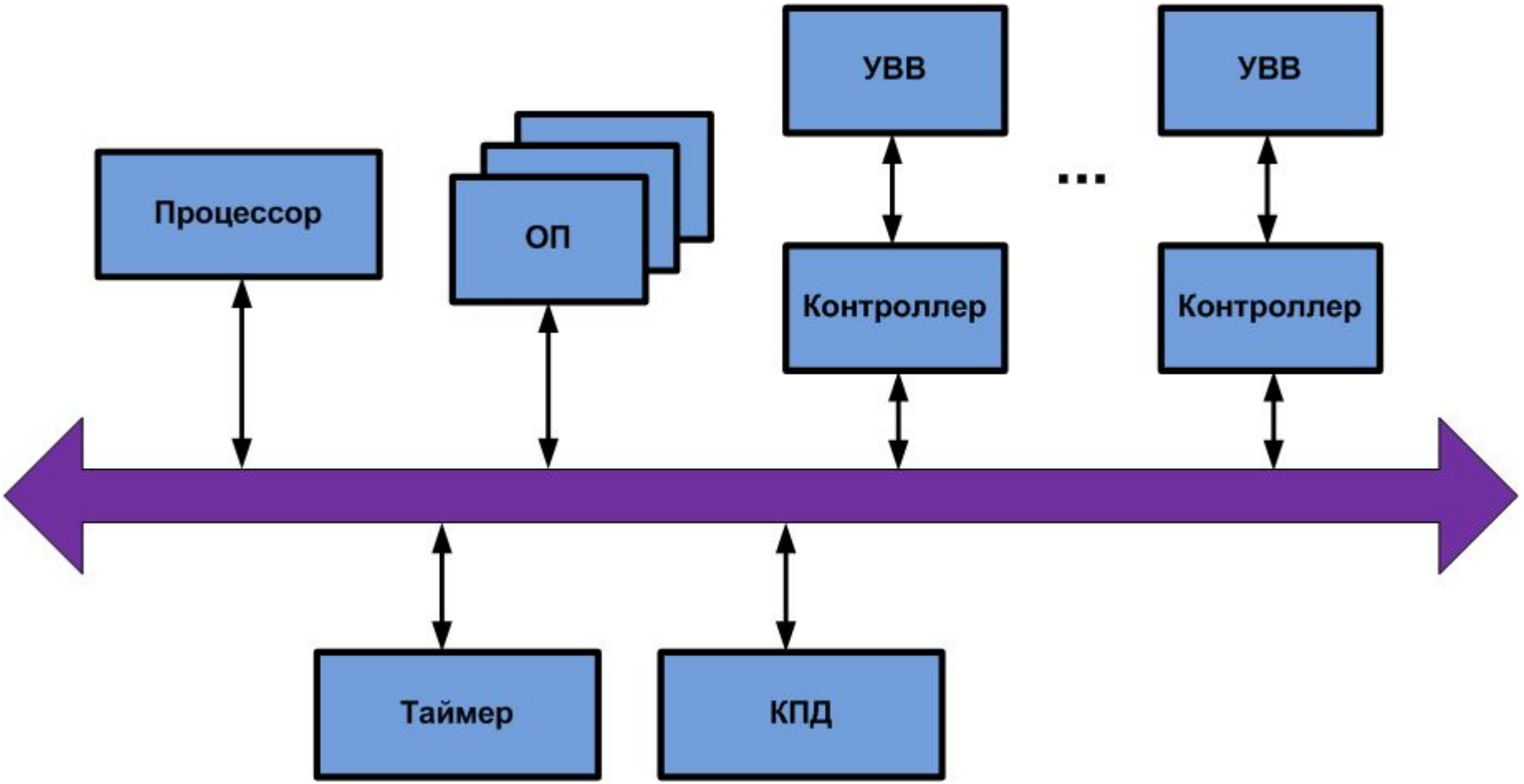


Рисунок 1.23– Общая структура ЭВМ

Понятие архитектуры процессора. Основные виды архитектур

Архитектурой процессора называется комплекс его аппаратных и программных средств, предоставляемых пользователю.

В это общее понятие входит набор программно-доступных регистров и исполнительных (операционных) устройств, система основных команд и способов адресации, объем и структура адресуемой памяти, виды и способы обработки прерываний.

При описании архитектуры и функционирования процессора обычно используется его представление в виде совокупности программно-доступных регистров, образующих **регистровую** или **программную модель**.

Регистр — последовательностное логическое устройство, используемое для хранения ***n***-разрядных двоичных чисел и выполнения преобразований над ними.

Регистр представляет собой упорядоченную последовательность триггеров, число которых соответствует числу разрядов в слове. С каждым регистром обычно связано комбинационное цифровое устройство, с помощью которого обеспечивается выполнение некоторых операций над словами.

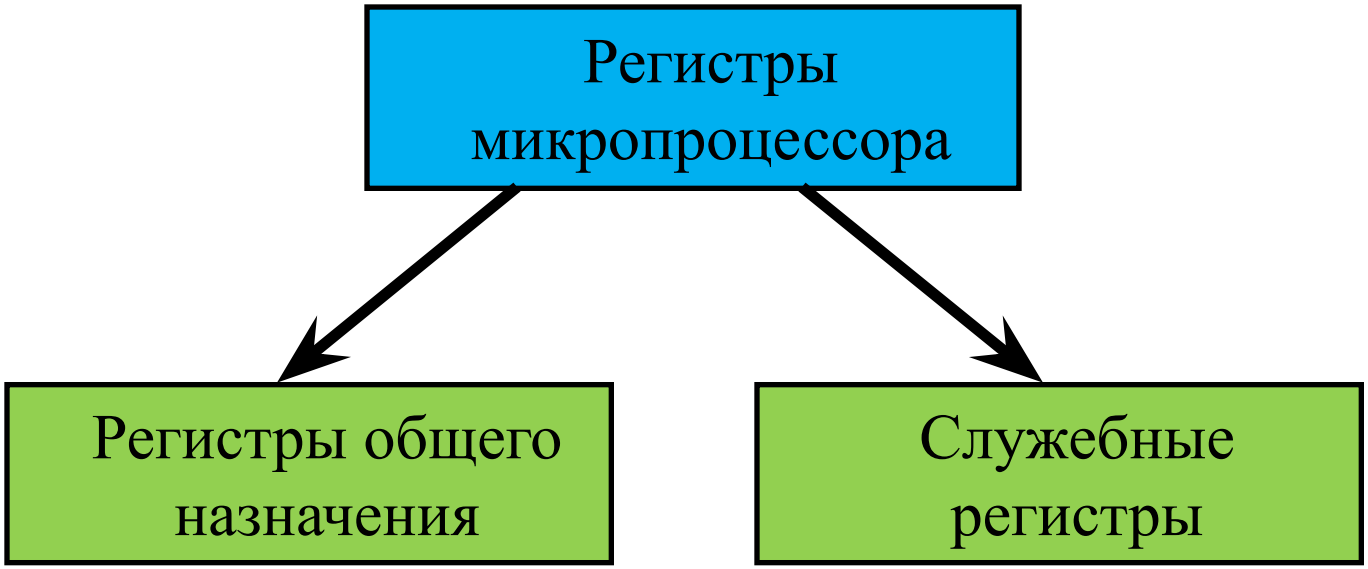


Рисунок 1.4 -Классификация регистров МП

Регистры общего назначения служат для хранения операндов.

Служебные регистры обеспечивают управление выполнением программы и режимом работы процессора, организацию обращения к памяти (защита памяти, сегментная и страничная организация и др.)

Состав и количество **служебных регистров** определяется архитектурой микропроцессора.

Обычно в их состав входят:

- программный счетчик PC (или CS + IP в архитектуре микропроцессоров Intel);
- регистр состояния SR (или EFLAGS);
- регистры управления режимом работы процессора CR (Control Register);
- регистры, реализующие сегментную и страничную организацию памяти;
- регистры, обеспечивающие отладку программ и тестирование процессора.

Функционирование процессора представляется в виде реализации регистровых пересылок - процедур изменения состояния этих регистров путем чтения-записи их содержимого.

CISC(Complex Instruction Set Computer)- архитектура реализована во многих типах микропроцессоров, выполняющих большой набор разноформатных команд с использованием многочисленных способов адресации.

Характеризуется следующим набором свойств:

- нефиксированным значением длины команды;

- исполнение операций, таких как загрузка в память, арифметические действия кодируется в одной инструкции;

-небольшим числом регистров, каждый из которых выполняет строго определённую функцию.

Типичными представителями являются процессоры на основе x86 команд (исключая современные Intel Pentium 4, Pentium D, Core, AMD Athlon, Phenom которые являются гибридными).

RISC(Reduced Instruction Set Computer) - архитектура отличается использованием ограниченного набора команд фиксированного формата Характерные особенности RISC-процессоров:

-фиксированная длина машинных инструкций (например, 32 бита) и простой формат команды;

- одна инструкция выполняет только одну операцию с памятью — чтение или запись. Операции вида «прочитать-изменить-записать» отсутствуют;

- большое количество регистров общего назначения (32 и более).

1.3 Понятие архитектуры процессора. Основные виды архитектур

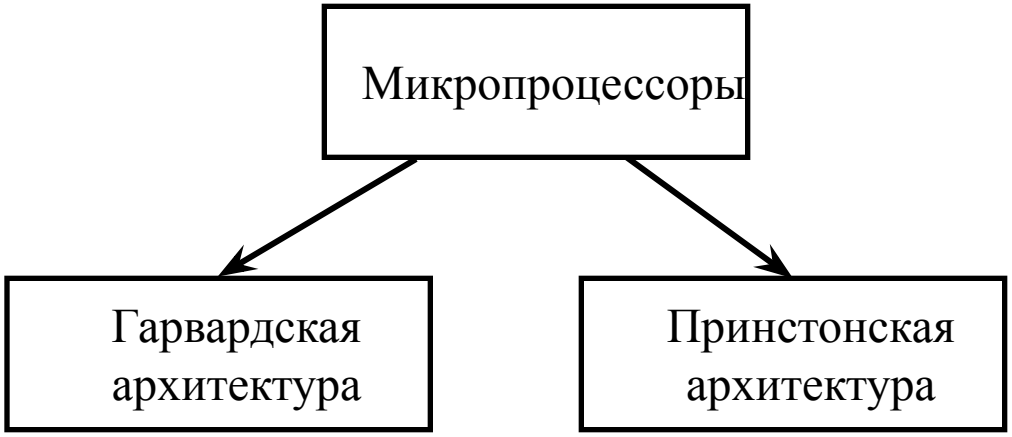


Рис.5 – Классификация МП по реализации памяти

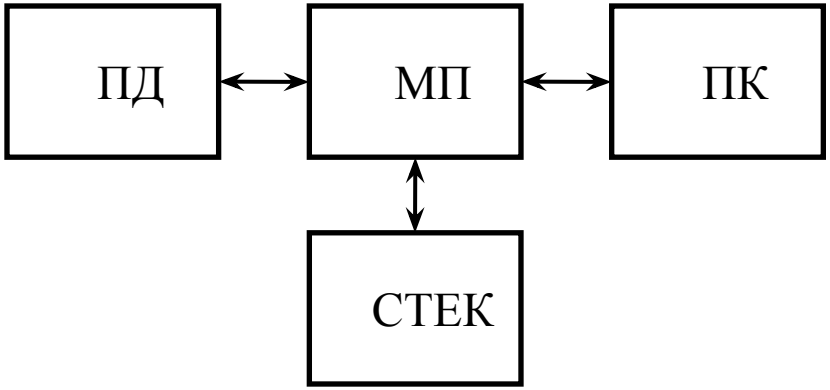


Рис.6 – Гарвардская архитектура

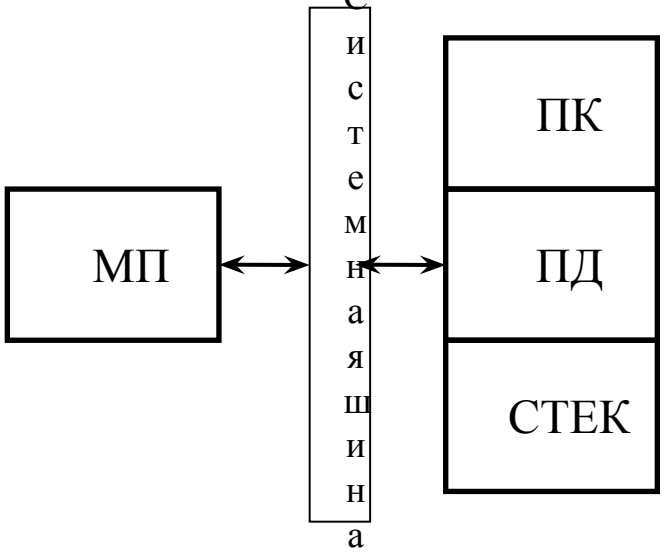
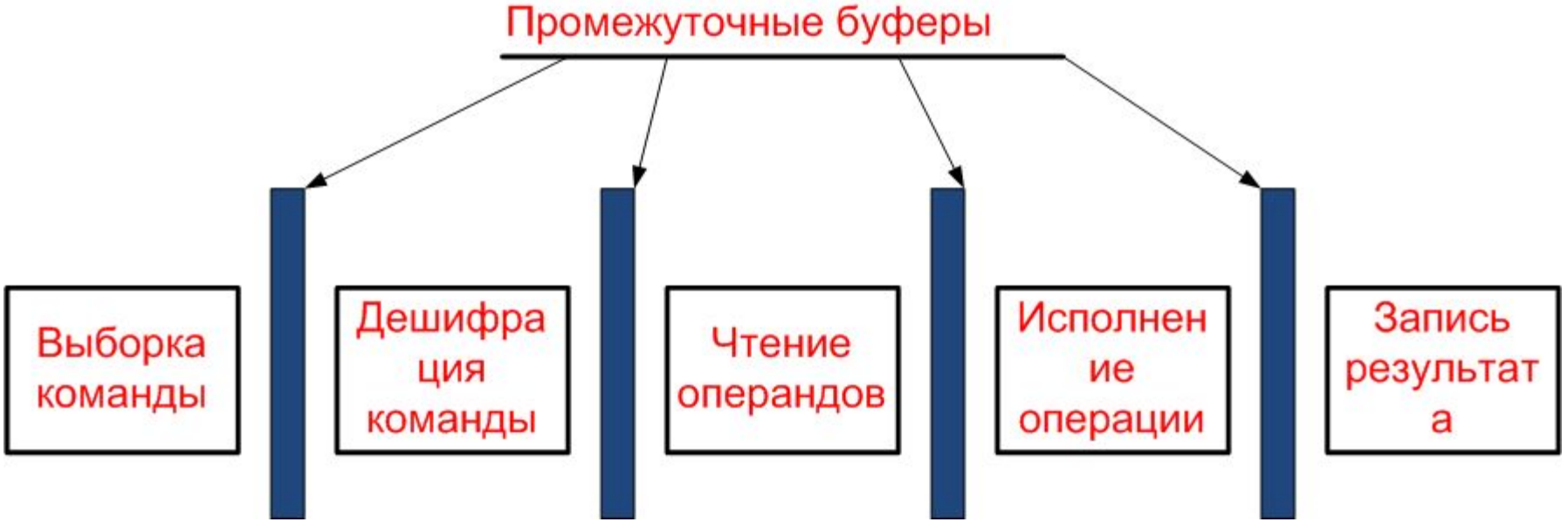


Рис. 7 – Принстонская архитектура

Конвейерная организация работы процессора



Такты работы процессора

	1	2	3	4	5	6	7	8	9	10
Команда *i*	IF	ID	RD	EX	WB					
Команда *i*+1		IF	ID	RD	EX	WB				
Команда *i*+2			IF	ID	RD	EX	WB			
Команда *i*+3				IF	ID	RD	EX	WB		
Команда *i*+4					IF	ID	RD	EX	WB	
Команда *i*+5						IF	ID	RD	EX	WB

Рис. 8 – Конвейерная обработка

Лабораторная работа №1

- **Методический материал**

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Оренбургский государственный университет»

Кафедра вычислительной техники

Р.Р. Галимов

**АППАРАТНЫЕ СРЕДСТВА
ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**

Рекомендовано к изданию Редакционно-издательским советом
федерального государственного бюджетного образовательного учреждения
высшего профессионального образования
«Оренбургский государственный университет»
в качестве методических указаний для студентов, обучающихся по программам
высшего профессионального образования по направлению подготовки
090900.62 Информационная безопасность

Оренбург
2012

Регистр флагов микропроцессора

Цель работы: изучить назначение регистра флагов микропроцессора 8086.

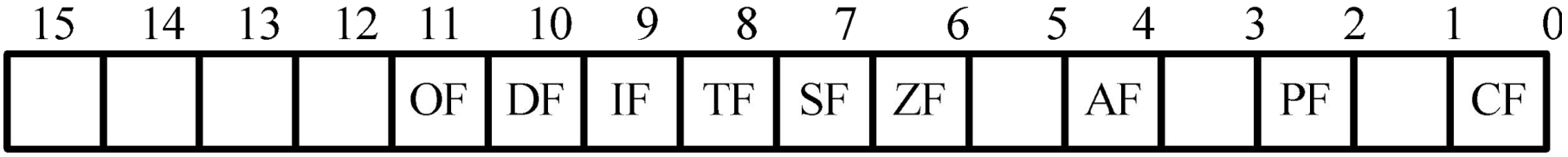
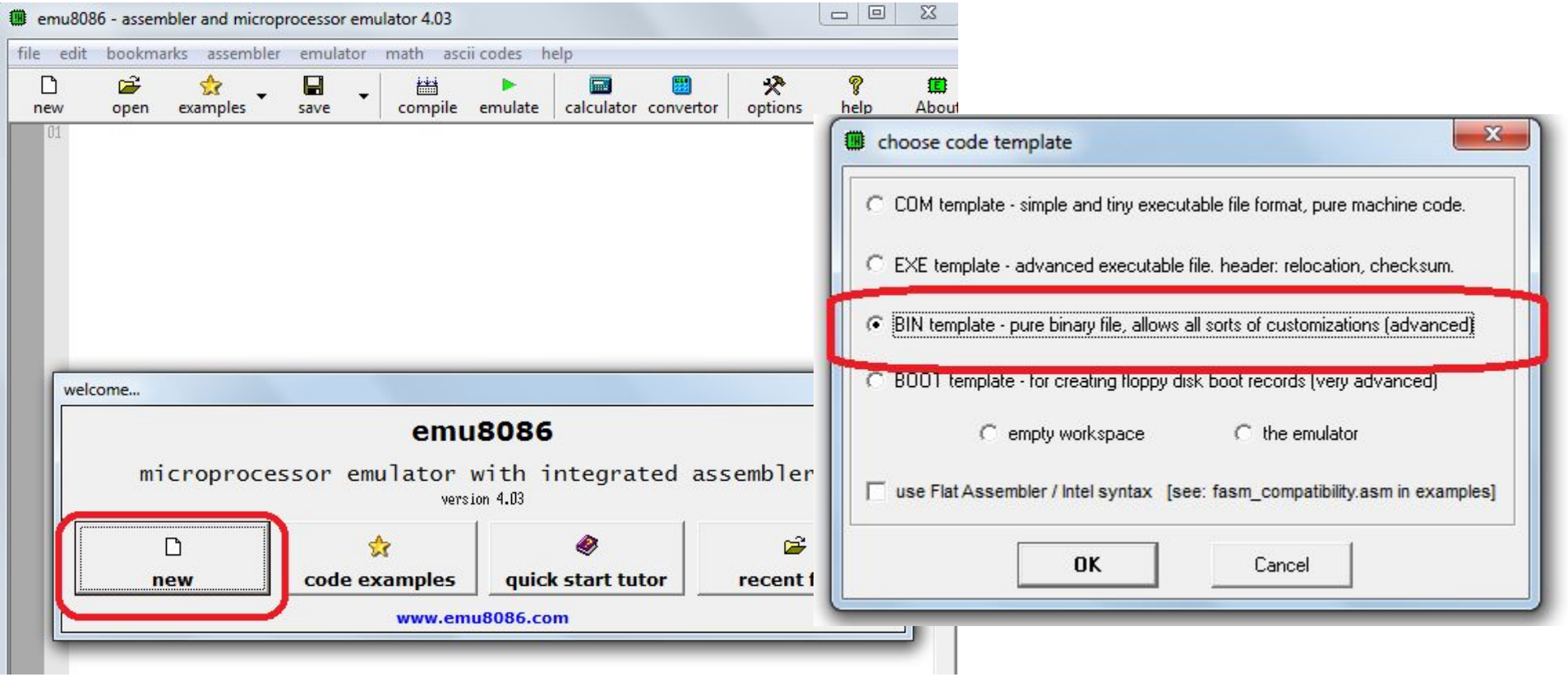
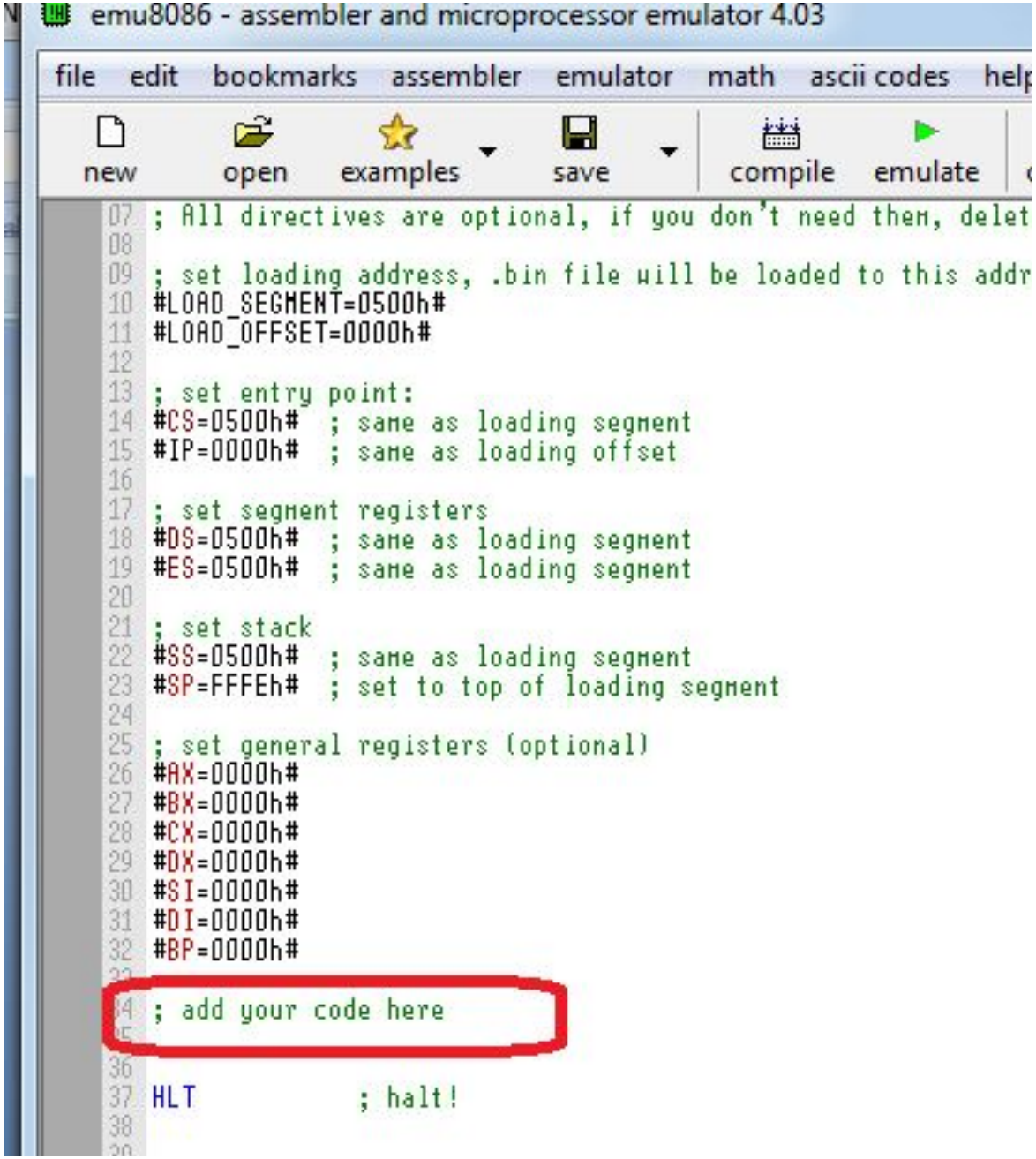
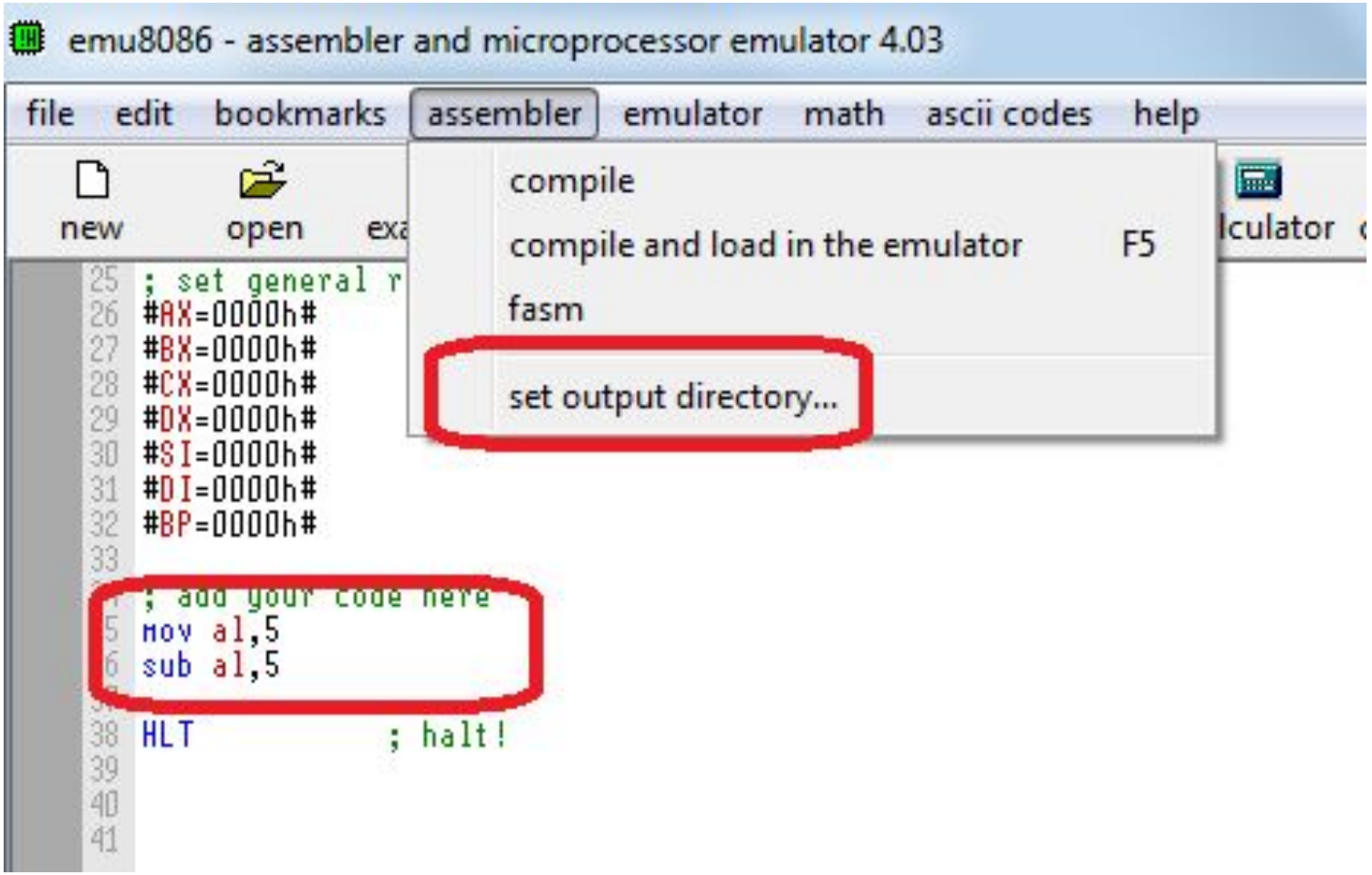
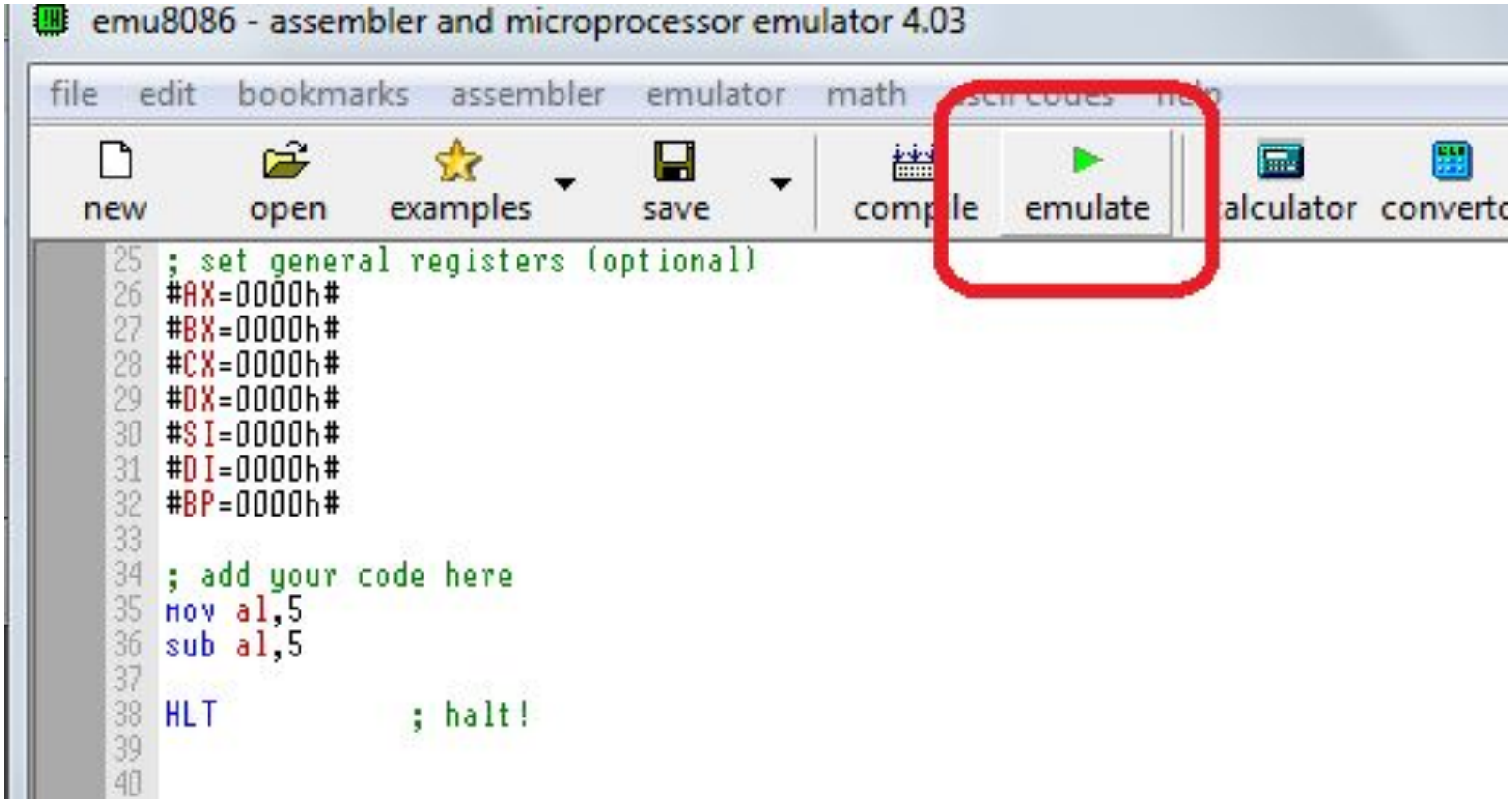


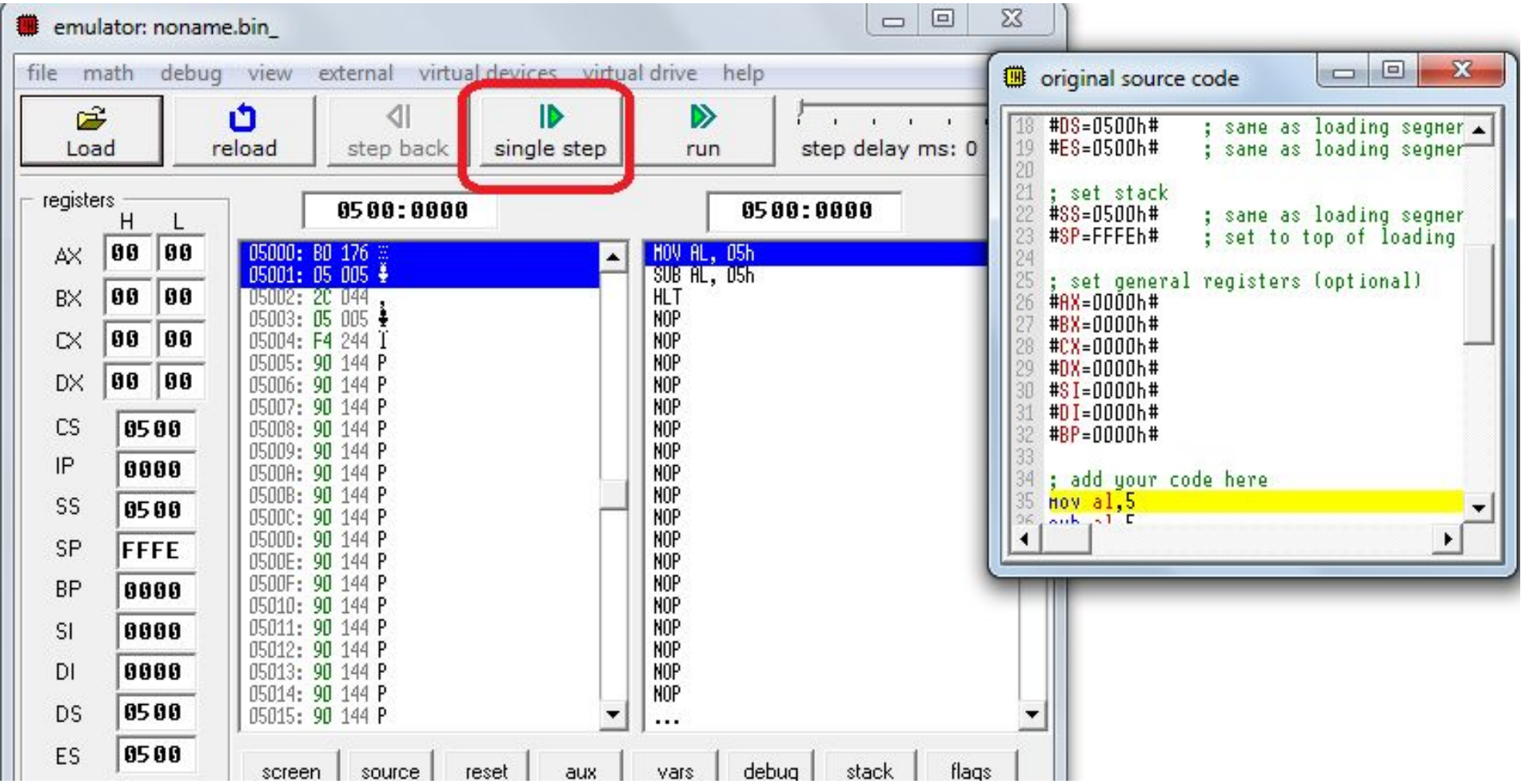
Рисунок 1.1 - Регистр флагов МП 8086

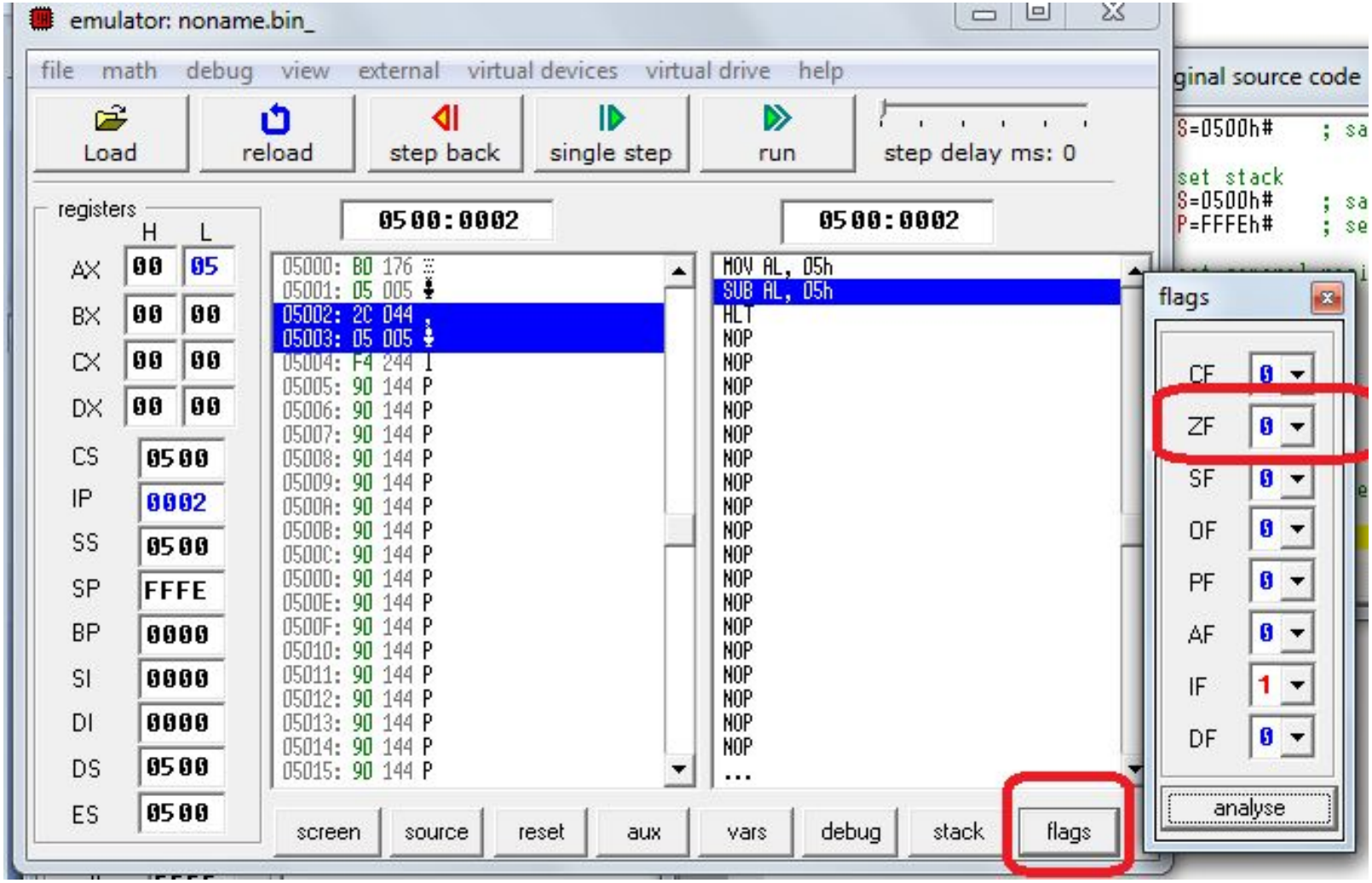

emu8086 - assembler and microprocessor emulator 4.03
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help About
01
welcome...
emu8086
microprocessor emulator with integrated assembler
version 4.03
new
code examples
quick start tutor
recent f
www.emu8086.com
choose code template
COM template - simple and tiny executable file format, pure machine code.
EXE template - advanced executable file. header: relocation, checksum.
BIN template - pure binary file, allows all sorts of customizations (advanced)
BOOT template - for creating floppy disk boot records (very advanced)
empty workspace
the emulator
use Flat Assembler / Intel syntax [see: fasm_compatibility.asm in examples]
OK
Cancel

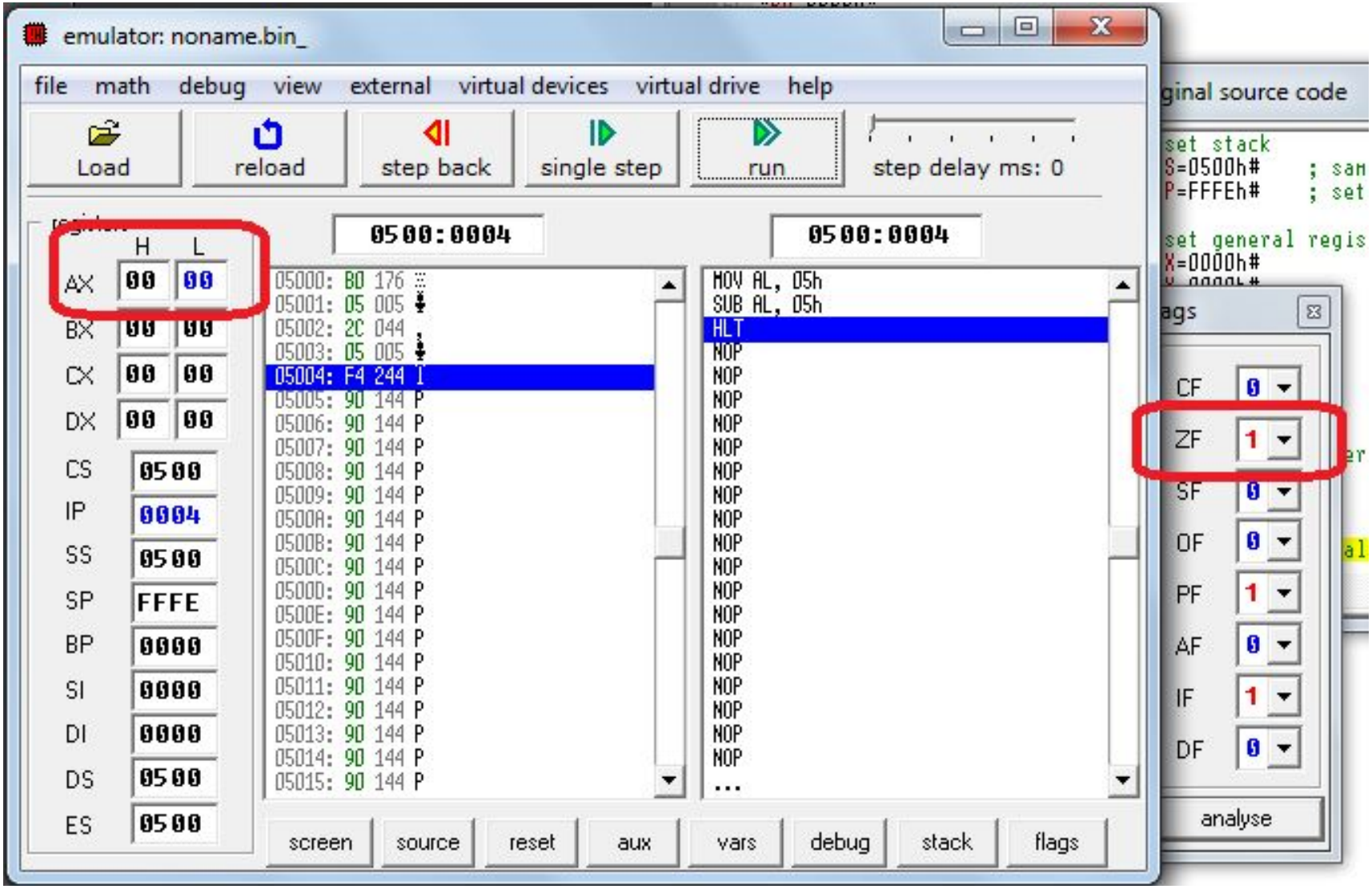

emu8086 - assembler and microprocessor emulator 4.03
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate
; All directives are optional, if you don't need them, delet
; set loading address, .bin file will be loaded to this addr
#LOAD_SEGMENT=0500h#
#LOAD_OFFSET=0000h#
; set entry point:
#CS=0500h# ; same as loading segment
#IP=0000h# ; same as loading offset
; set segment registers
#DS=0500h# ; same as loading segment
#ES=0500h# ; same as loading segment
; set stack
#SS=0500h# ; same as loading segment
#SP=FFFEh# ; set to top of loading segment
; set general registers (optional)
#AX=0000h#
#BX=0000h#
#CX=0000h#
#DX=0000h#
#SI=0000h#
#DI=0000h#
#BP=0000h#
; add your code here
HLT ; halt!


emu8086 - assembler and microprocessor emulator 4.03
file
edit
bookmarks
assembler
emulator
math
ascii codes
help
new
open
compile
compile and load in the emulator
F5
fasm
set output directory...
lculator
; set general r
#AX=0000h#
#BX=0000h#
#CX=0000h#
#DX=0000h#
#SI=0000h#
#DI=0000h#
#BP=0000h#
; add your code here
mov al,5
sub al,5
HLT ; halt!


emu8086 - assembler and microprocessor emulator 4.03
file edit bookmarks assembler emulator math
new open examples save emulate
; set general registers (optional)
#AX=0000h#
#BX=0000h#
#CX=0000h#
#DX=0000h#
#SI=0000h#
#DI=0000h#
#BP=0000h#
; add your code here
mov al,5
sub al,5
HLT ; halt!


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 00
BX 00 00
CX 00 00
DX 00 00
CS 0500
IP 0000
SS 0500
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0500
ES 0500
0500:0000
0500:0000
05000: B0 176
05001: 05 005
05002: 2C 044 ,
05003: 05 005
05004: F4 244 ô
05005: 90 144 É
05006: 90 144 É
05007: 90 144 É
05008: 90 144 É
05009: 90 144 É
0500A: 90 144 É
0500B: 90 144 É
0500C: 90 144 É
0500D: 90 144 É
0500E: 90 144 É
0500F: 90 144 É
05010: 90 144 É
05011: 90 144 É
05012: 90 144 É
05013: 90 144 É
05014: 90 144 É
05015: 90 144 É
MOV AL, 05h
SUB AL, 05h
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen source reset aux vars debug stack flags
original source code
18 #DS=0500h# ; same as loading segmer
19 #ES=0500h# ; same as loading segmer
20
21 ; set stack
22 #SS=0500h# ; same as loading segmer
23 #SP=FFFEh# ; set to top of loading
24
25 ; set general registers (optional)
26 #AX=0000h#
27 #BX=0000h#
28 #CX=0000h#
29 #DX=0000h#
30 #SI=0000h#
31 #DI=0000h#
32 #BP=0000h#
33
34 ; add your code here
35 mov al,5


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 05
BX 00 00
CX 00 00
DX 00 00
CS 0500
IP 0002
SS 0500
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0500
ES 0500
0500:0002
0500:0002
05000: B0 176
05001: 05 005
05002: 2C 044
05003: 05 005
05004: F4 244
05005: 90 144 P
05006: 90 144 P
05007: 90 144 P
05008: 90 144 P
05009: 90 144 P
0500A: 90 144 P
0500B: 90 144 P
0500C: 90 144 P
0500D: 90 144 P
0500E: 90 144 P
0500F: 90 144 P
05010: 90 144 P
05011: 90 144 P
05012: 90 144 P
05013: 90 144 P
05014: 90 144 P
05015: 90 144 P
MOV AL, 05h
SUB AL, 05h
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen
source
reset
aux
vars
debug
stack
flags
flags
CF 0
ZF 0
SF 0
OF 0
PF 0
AF 0
IF 1
DF 0
analyse
ginal source code
S=0500h# ; sa
set stack
S=0500h# ; sa
P=FFFEh# ; se


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
H L
AX 00 00
BX 00 00
CX 00 00
DX 00 00
CS 0500
IP 0004
SS 0500
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0500
ES 0500
0500:0004
0500:0004
05000: B0 176 °
05001: 05 005
05002: 2C 044 ,
05003: 05 005
05004: F4 244 ⌠
05005: 90 144 É
05006: 90 144 É
05007: 90 144 É
05008: 90 144 É
05009: 90 144 É
0500A: 90 144 É
0500B: 90 144 É
0500C: 90 144 É
0500D: 90 144 É
0500E: 90 144 É
0500F: 90 144 É
05010: 90 144 É
05011: 90 144 É
05012: 90 144 É
05013: 90 144 É
05014: 90 144 É
05015: 90 144 É
MOV AL, 05h
SUB AL, 05h
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen
source
reset
aux
vars
debug
stack
flags
ginal source code
set stack
S=0500h#
P=FFFEh#
set general regis
X=0000h#
ags
CF 0
ZF 1
SF 0
OF 0
PF 1
AF 0
IF 1
DF 0
analyse

```
CMP AL,100 ; устанавливаем флаги
JA NEXT ; если AL>100, то переходим на адрес, указанный меткой
    next
MOV AH,2 ; иначе умножаем содержимое регистра AL на 2
MUL AH
JMP exit ; переход на метку exit
next: ADD AL,5 ; прибавляем к содержимому AL число 5
exit:
hlt
```

Таблица 1.1 –Команды условных переходов

Код команды	Реальное условие	Условие для перехода (если)
JA ***JBE***	***CF*** = 0 и ***ZF*** = 0	Выше Не ниже и не равно
JAE ***JNB*** ***JNC***	***CF*** = 0	Выше или равно Не ниже Нет переноса
JB ***JNAE*** ***JC***	***CF***=1	Ниже Не выше и не равно Перенос
JBE ***JNA***	***CF*** = 1 или ***ZF*** = 1	Ниже или равно Если не выше
JE ***JZ***	***ZF*** = 1	Равно Ноль
JG ***JNLE***	***ZF*** = 0 и ***SF*** = ***OF***	Больше Не меньше и не равно
JGE ***JNL***	***SF*** = ***OF***	Больше или равно Не меньше
JL ***JNGE***	***SF***<>***OF***	Меньше Не больше и не равно
JLE ***JNG***	***ZF*** = 1 или (***SF xor OF***)=1	Меньше или равно Не больше
JNE ***JNZ***	***ZF*** = 0	Не равно Не ноль
JNO	***OF*** = 0	Нет переполнения
JO	***OF***=1	Есть переполнение
JNP ***JPO***	***PF*** = 0	Нет четности Нечетное
JP ***JPE***	***PF***=1	Есть четность Четное
JNS	***SF*** = 0	Нет знака

Вопросы для самопроверки

1. Преимущества цифровой обработки информации?
2. Основные принципы Фон Неймана
3. Дайте определение микропроцессора и микропроцессорной системы
4. Что понимается под основной памятью?
5. Принцип модульности построения МПС?
6. Принцип магистральности построения МПС?
7. Дайте определение архитектуры микропроцессора
8. Что такое регистр микропроцессора?
9. Приведите примеры служебных регистров микропроцессора
10. Перечислите характеристики CISC-процессоров
11. Какой класс процессоров обычно реализуются по Гарвардской архитектуре
12. В чем преимущества конвейерной обработки команд?