Условные переходы

В микропроцессорах x86 и x64 помимо регистров (eax, ebx, ecx…) применяемых для реализации арифметических и логических команд, есть ещё один очень важный регистр флагов процессора.
Регистр флагов – используется при выполнении большинства команд. Регистр флагов носит название **EFLAGS**. Это 32-разрядный регистр. Однако старшие 16 разрядов используются в защищённом режиме работы операционной системы, и пока мы их рассматривать не будем.

Каждый бит в регистре **FLAGS** является флагом. **Флаг** – это один или несколько битов памяти, которые могут принимать двоичные значения (или комбинации значений) и характеризуют состояние какого-либо объекта.

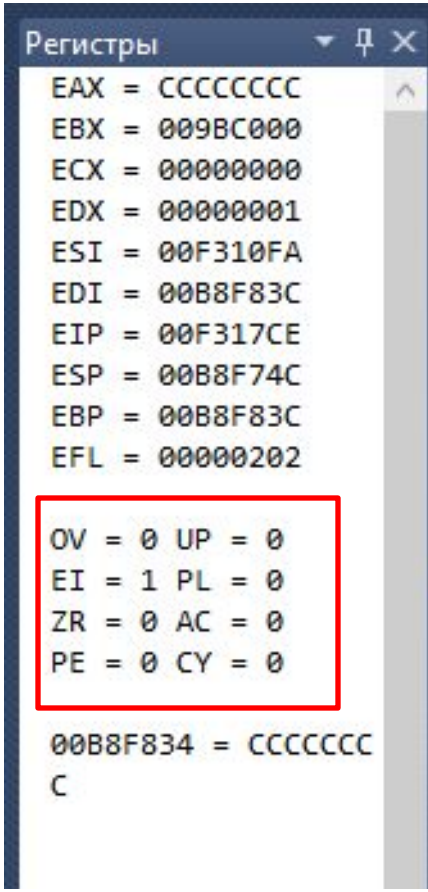
Регистр флагов содержит группу **флагов состояния**, **управляющий флаг** и **группу системных флагов**.

Регистр флагов

Бит	Обозначение	Название	Описание
0	CF	Carry Flag	**Флаг переноса**. Устанавливается в 1, если результат предыдущей операции не уместился в приёмнике и произошёл перенос из старшего бита или если требуется заём (при вычитании). Иначе установлен в 0.
1	1	-	Зарезервирован.
2	PF	Parity Flag	**Флаг чётности**. Устанавливается в 1, если младший байт результата предыдущей команды содержит чётное количество битов, равных 1. Если количество единиц в младшем байте нечётное, то этот флаг равен 0.
3	0	-	Зарезервирован.
4	AF	Auxiliary Carry Flag	**Вспомогательный флаг переноса** (или **флаг полупереноса**). Устанавливается в 1, если в результате предыдущей операции произошёл перенос (или заём) из третьего бита в четвёртый. Этот флаг используется автоматически командами двоично-десятичной коррекции.
5	0	-	Зарезервирован.
6	ZF	Zero Flag	**Флаг нуля**. Устанавливается 1, если результат предыдущей команды равен 0.
7	SF	Sign Flag	**Флаг знака**. Этот флаг всегда равен старшему биту результата.
8	TF	Trap Flag	**Флаг трассировки** (или **флаг ловушки**). Он был предусмотрен для работы отладчиков в пошаговом выполнении, которые не используют защищённый режим. Если этот флаг установить в 1, то после выполнения каждой программной команды управление временно передаётся отладчику (вызывается прерывание 1).
9	IF	Interrupt Enable Flag	**Флаг разрешения прерываний**. Если сбросить этот флаг в 0, то процессор перестанет обрабатывать прерывания от внешних устройств. Обычно его сбрасывают на короткое время для выполнения критических участков программы.
10	DF	Direction Flag	**Флаг направления**. Контролирует поведение команд обработки строк. Если установлен в 1, то строки обрабатываются в сторону уменьшения адресов, если сброшен в 0, то наоборот.
11	OF	Overflow Flag	**Флаг переполнения**. Устанавливается в 1, если результат предыдущей арифметической операции над числами со знаком выходит за допустимые для них пределы. Например, если при сложении двух положительных чисел получается число со старшим битом, равным единице, то есть отрицательное. И наоборот.
12 13	IOPL	I/O Privilege Level	Уровень приоритета ввода/вывода.
14	NT	Nested Task	Флаг вложенности задач.
15	0	-	Зарезервирован.

В **visual studio** текущее значение флагов можно посмотреть в окне **Регистры**. При этом нужно отметить, что обозначения флагов в **visual studio** существенно отличаются от общепринятых!!!

Flag	Значение
Переполнение	OV = 1
Направление (пока не рассматриваем!!)	UP = 1
Прерывание (пока не рассматриваем!!)	EI = 1
Флаг знака	PL = 1
Флаг нуля	ZR = 1
Добавочный перенос (пока не рассматриваем!!)	AC = 1
Четность (пока не рассматриваем!!)	PE = 1
Перенос / заём (при сложении / вычитании)	CY = 1



Проведите анализ программы:

Задания: Выполните отладку программы наблюдая за значениями переменных в окне **Локальные**, значением регистра **eax** и состояниями флагов **OV, PL, ZR, CY**. Пошагово выполняя программу проследите и отразите в отчете какие команды и инструкции изменяют флаги (какие именно флаги меняются при этом), объясните почему происходит изменение этих флагов анализируя содержимое переменных и регистра **eax**.

Составьте таблицу микропроцессорных кодов.

```
#include "stdafx.h"
#include <iostream>


using namespace std;

void main()
{
    int a=10;
    int b=10;
    int c;
    int d;

    c=a-b;
    c=c-1;
    d=2147483647;
    d++;
    c++;
    d--;
}
```