**Цель работы**

Целью работы является изучение системы команд микроконтроллеров семейства *Intel 8051*, также продолжение начатого в лабораторной работе № 1 изучения интегрированной среды *ProView* фирмы *Franklin Software Inc.*, которая предназначена для разработки программного обеспечения для этого семейства.

**Исследование примеров программ**

**Пример 1. Запись данных**

Записать в резидентную память данных по адресам *41* и *42* число *1C3FH*:

**START: JMP LOAD ; переход к программе**

**ORG 30H ; директива размещения программы с адр. 30h**

**LOAD: MOV R0, #41h ; загрузка в R0 указателя данных**

**MOV @R0, #1Ch ; запись в память числа 1Ch**

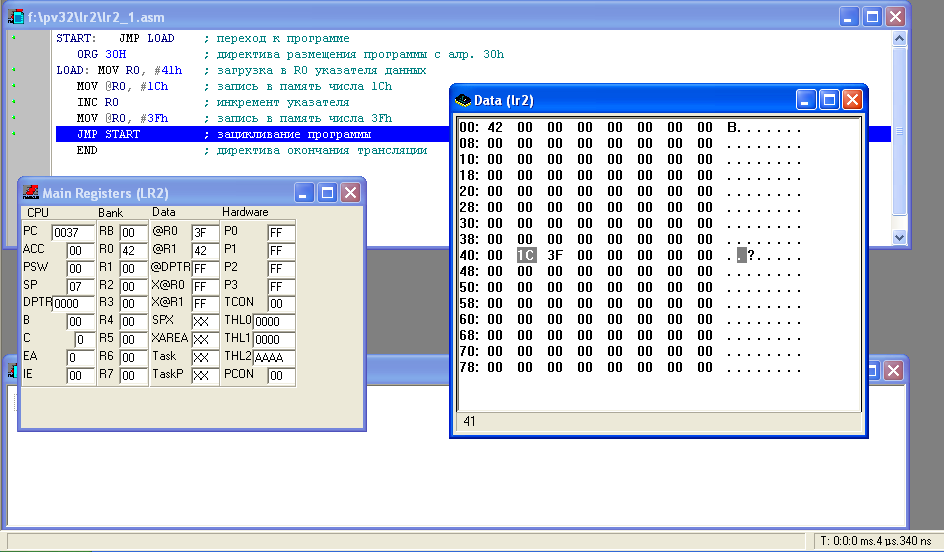
**INC R0 ; инкремент указателя**

**MOV @R0, #3Fh ; запись в память числа 3Fh**

**JMP START ; зацикливание программы**

**END ; директива окончания трансляции**

Результат выполнения программы:



**Пример 2. Сложение**

Сложение многобайтных чисел, *R*0 и *R1* – указатели на начальные адреса (младшие байты) первого и второго слагаемых соответственно, *R2* – длина операндов в байтах, сумма помещается на место первого слагаемого.

При сложении чисел без знака на переполнение укажет флаг *C (CY)*, а в случае сложения число со знаком — флаг *OV*.

**START: JMP ADDM ; переход к программе**

**ORG 30H ; директива размещения программы с адр. 30h**

**ADDM: MOV R0, #30h ; загрузка нач. адр. 1-го слагаемого и результата**

**MOV R1, #34h ; загрузка нач. адр. второго слагаемого**

**MOV R2, #04h ; длина операндов**

**CLR C ; сброс переноса**

**LOOP: MOV A, @R0 ; загрузка в A текущ. байта первого слагаемого**

**ADDC A, @R1 ; сложение байтов с учетом переноса**

**MOV @R0, A ; запись байта результата**

**INC R0 ; продвижение указателей**

**INC R1**

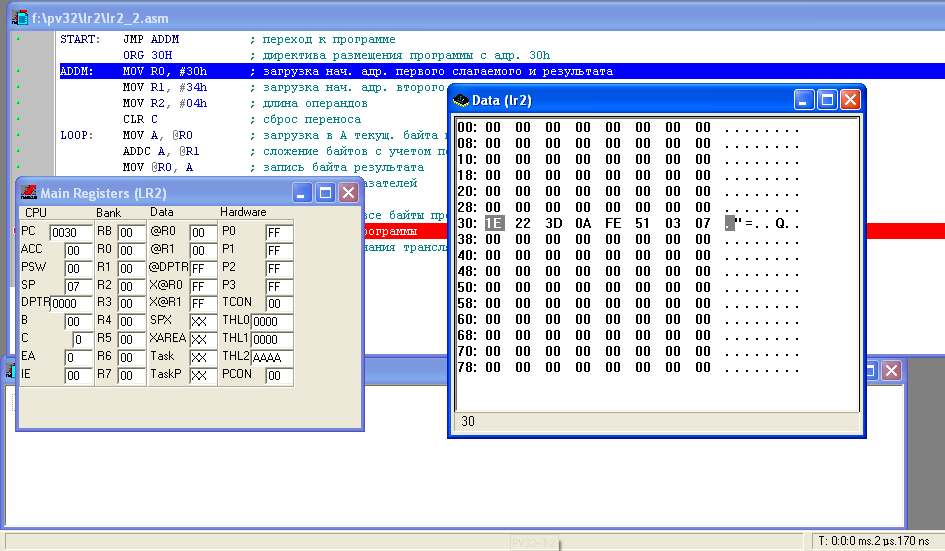
**DJNZ R2, LOOP ; цикл, если не все байты просуммированы**

**JMP START ; зацикливание программы**

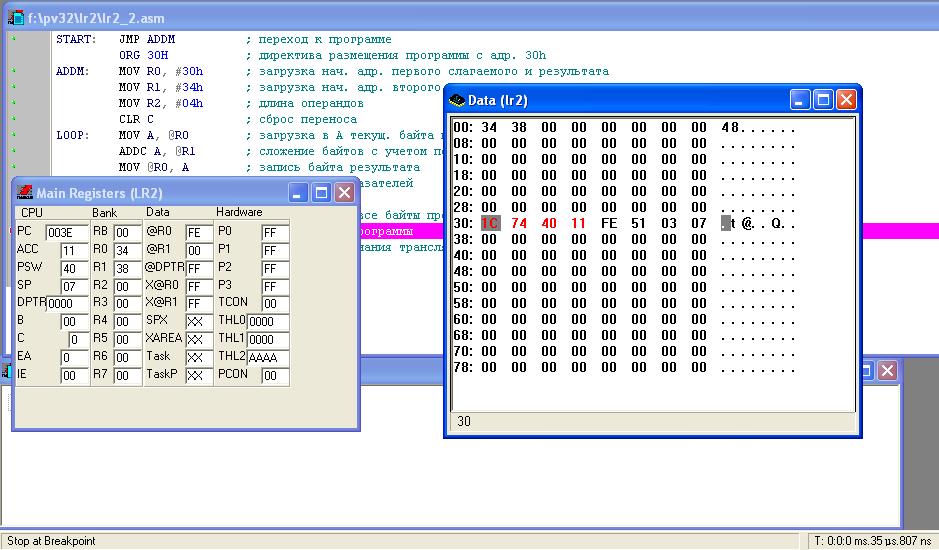
**END ; директива окончания трансляции**

Результат выполнения программы:

а) начальные значения операндов: первый операнд адреса 30h-33h, второй – 34h-37h



б) результат – сумма – адреса 30-33h



**Пример 3. Умножение**

Пусть требуется умножить целое двоичное число произвольного формата на константу *73.* Исходное число размешается в резидентной памяти данных, адрес младшего байта находится в регистре *R0*. Формат числа в байтах хранится в *R1*.

Полученное произведение размещается на месте исходного числа и занимает в памяти на один байт больше.

**START: JMP MULK ; переход к программе**

**ORG 30H ; директива размещения программы с адр. 30h**

**MULK: MOV R0, #30h ; загрузка нач. адр. множимого**

**MOV R1, #03h ; формат числа в байтах**

**CLR A ; сброс аккумулятора**

**LOOP: XCH A, @R0 ; загрузка множимого**

**MOV B, #73 ; загрузка множителя**

**MUL AB ; умножение**

**ADD A, @R0 ; прибавляем старш. часть от предыд. част. произв.**

**MOV @R0, A ; запись младшего байта частичного произведения**

**INC R0 ; приращение указателя**

**MOV A, B ; пересылка старш. байта частичного произв. в A**

**DJNZ R1, LOOP ; цикл, если не все байты умножены**

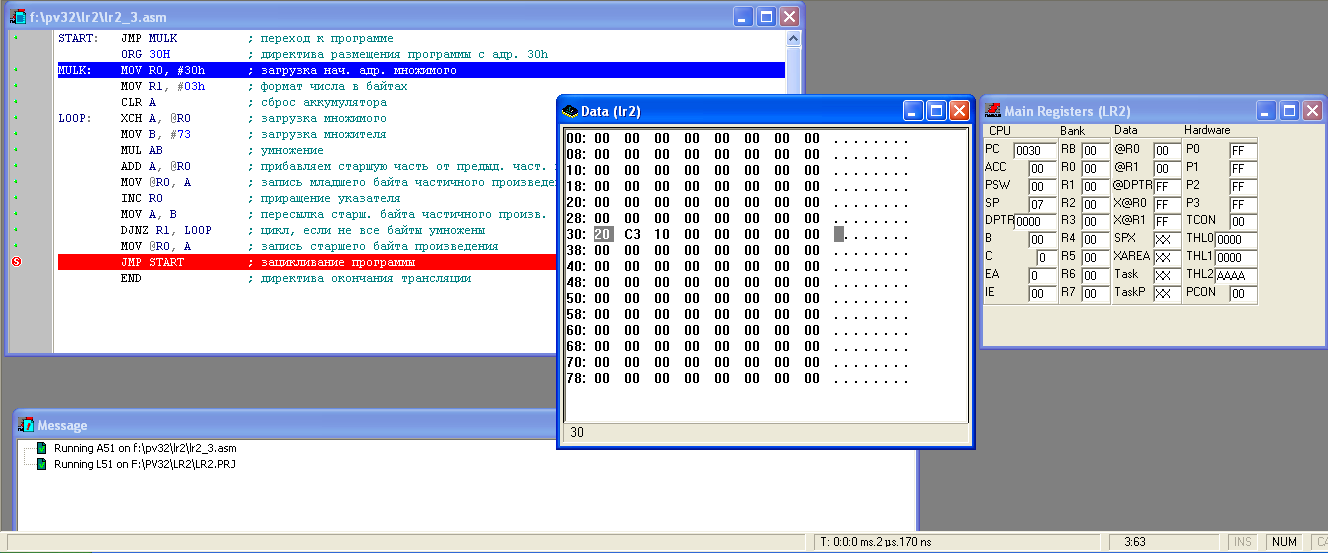
**MOV @R0, A ; запись старшего байта произведения**

**JMP START ; зацикливание программы**

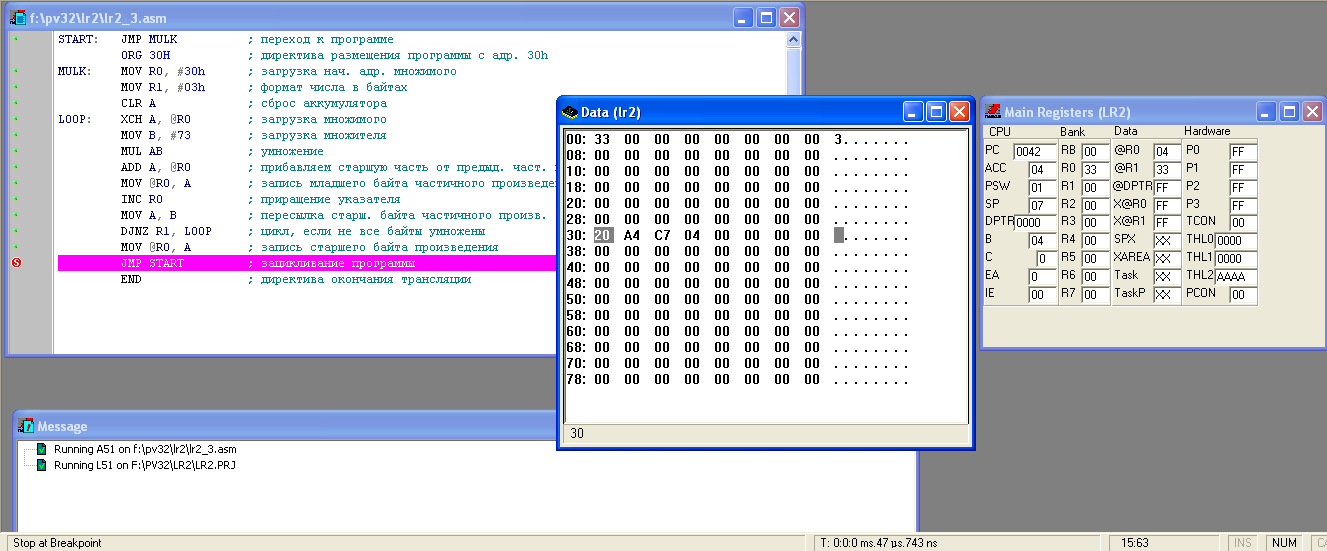
**END ; директива окончания трансляции**

Результат выполнения программы:

а) значение множимого адреса 30h-32h



б) результат – произведение на константу 73 – адреса 30h-33h



**Пример 4. Деление**

В качестве примера рассмотрим программу, которая переводит двоичное число, содержащееся в аккумуляторе, в *BCD*-код. При таком преобразовании может получиться трехразрядное *BCD*-число. Старшая цифра (число сотен) будет размещена в регистре *R0*, а две младшие в аккумуляторе:

**START: JMP MULK ; переход к программе**

**ORG 30H ; директива размещения программы с адр. 30h**

**MULK: MOV R0, #30h ; загрузка нач. адр. множимого**

**MOV R1, #03h ; формат числа в байтах**

**CLR A ; сброс аккумулятора**

**LOOP: XCH A, @R0 ; загрузка множимого**

**MOV B, #73 ; загрузка множителя**

**MUL AB ; умножение**

**ADD A, @R0 ; прибавляем старш. часть от предыд. част. произв.**

**MOV @R0, A ; запись младшего байта частичного произведения**

**INC R0 ; приращение указателя**

**MOV A, B ; пересылка старш. байта частичного произв. в A**

**DJNZ R1, LOOP ; цикл, если не все байты умножены**

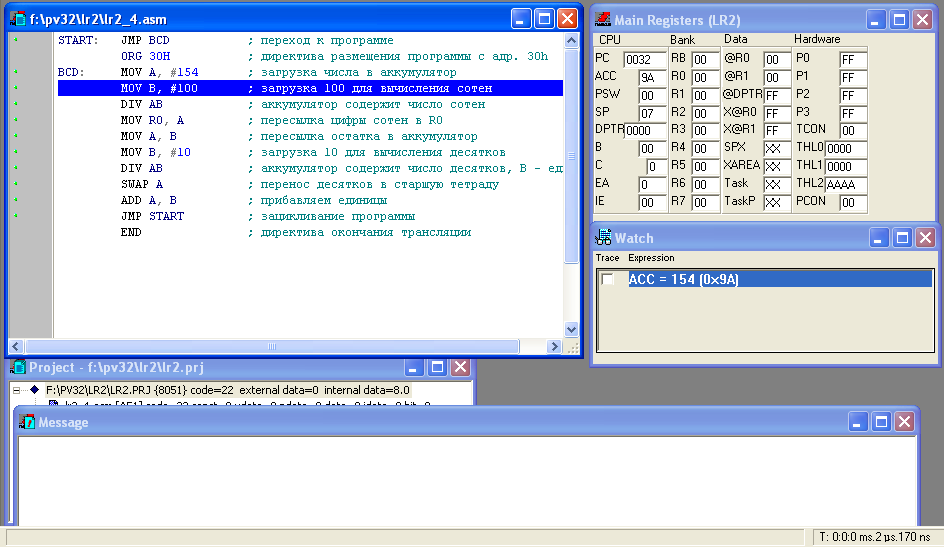
**MOV @R0, A ; запись старшего байта произведения**

**JMP START ; зацикливание программы**

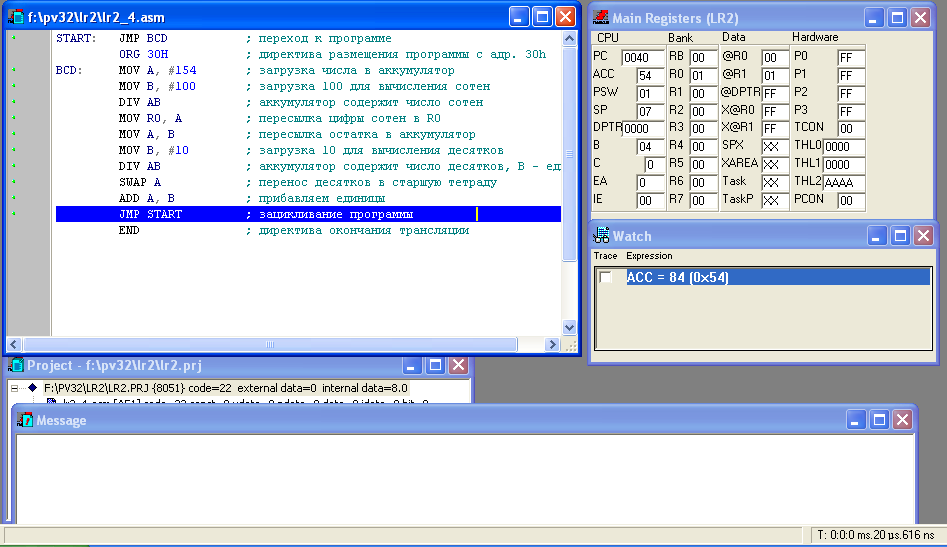
**END ; директива окончания трансляции**

Результат выполнения программы:

а) значение исходного числа в A = 154



б) результат перевода: сотни в R0, десятки и единицы в A



**Пример 5. Операции со стеком**

Подпрограмма должна сохранить в стеке содержимое тех регистров, которые она сама будет использовать, а перед возвратом в прерванную программу должна восстановить их значения

Подпрограмма с дополнениями для ее тестирования может, например, иметь следующую структуру:

**START: MOV R1, #02h ; загрузка регистров**

**MOV A, #30h**

**MOV R2 #00h**

**LCALL SUB ; переход на подпрограмму**

**SJMP START**

**SUB: PUSH PSW ; сохранение в стеке PSW**

**PUSH ACC ; сохранение в стеке аккумулятора**

**PUSH B ; сохранение в стеке расширителя аккумулятора B**

**ADD A, R1 ; некоторая обработка данных**

**MOV R2, A**

**POP B ; восстановление B**

**POP ACC ; восстановление A**

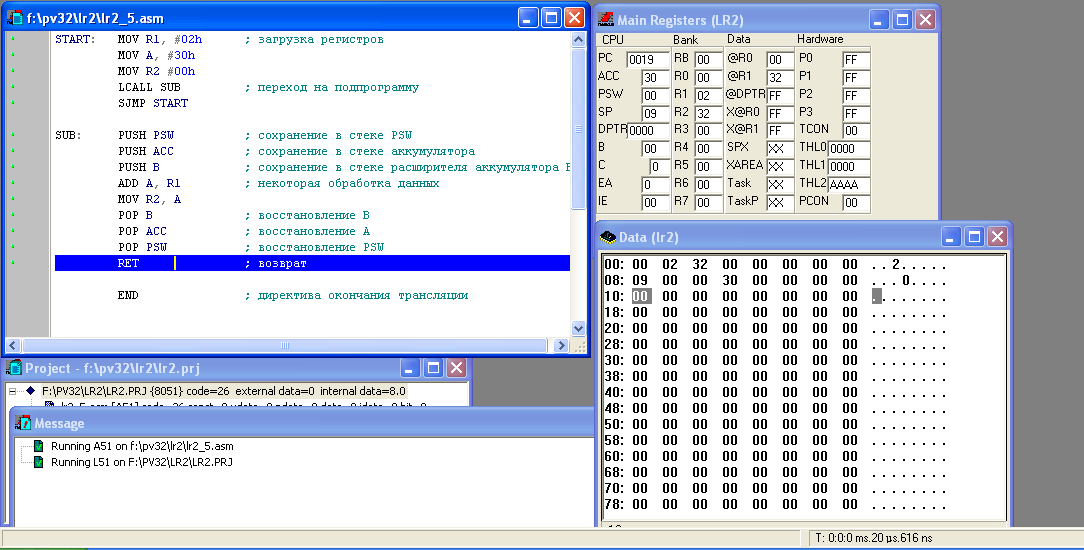
**POP PSW ; восстановление PSW**

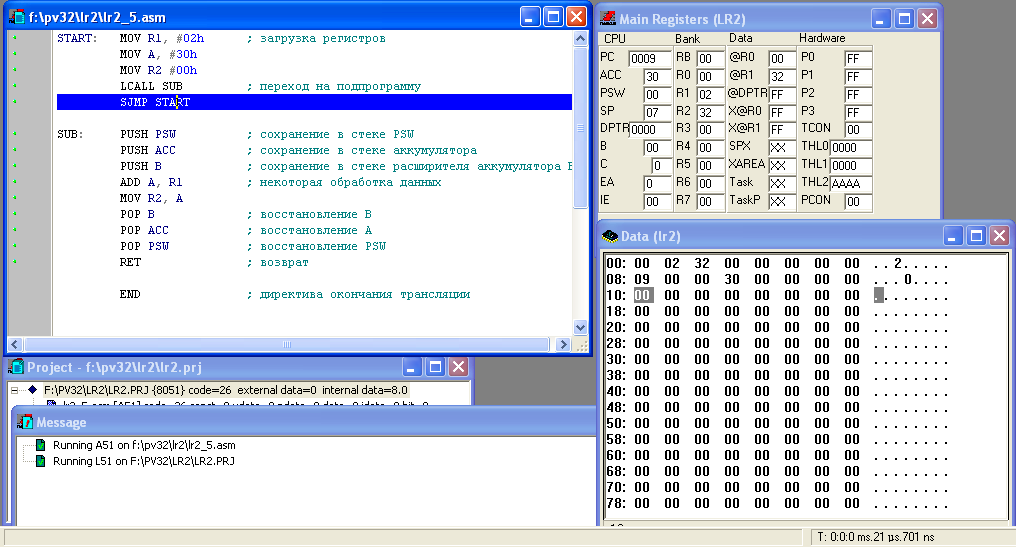
**RET ; возврат**

**END ; директива окончания трансляции**

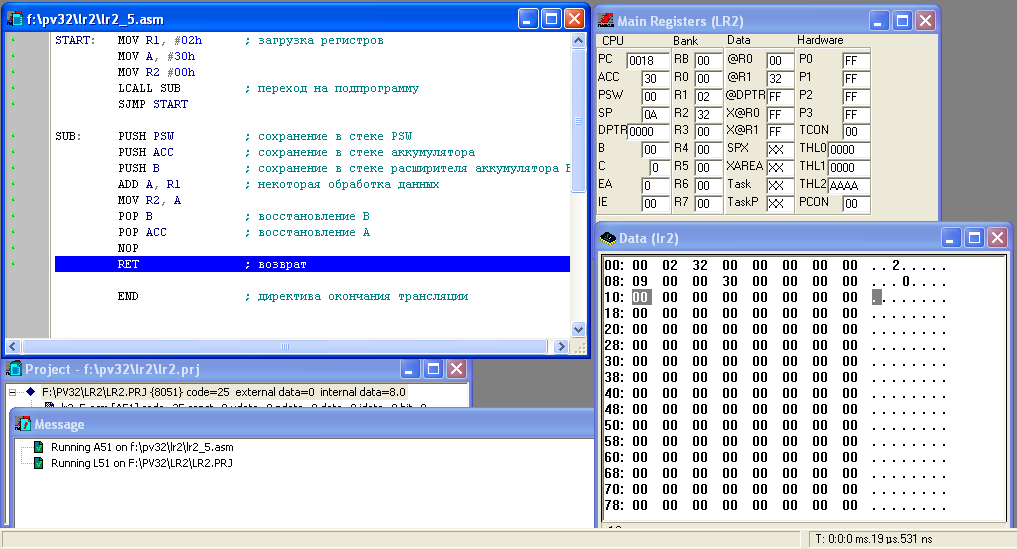
Результат выполнения программы:

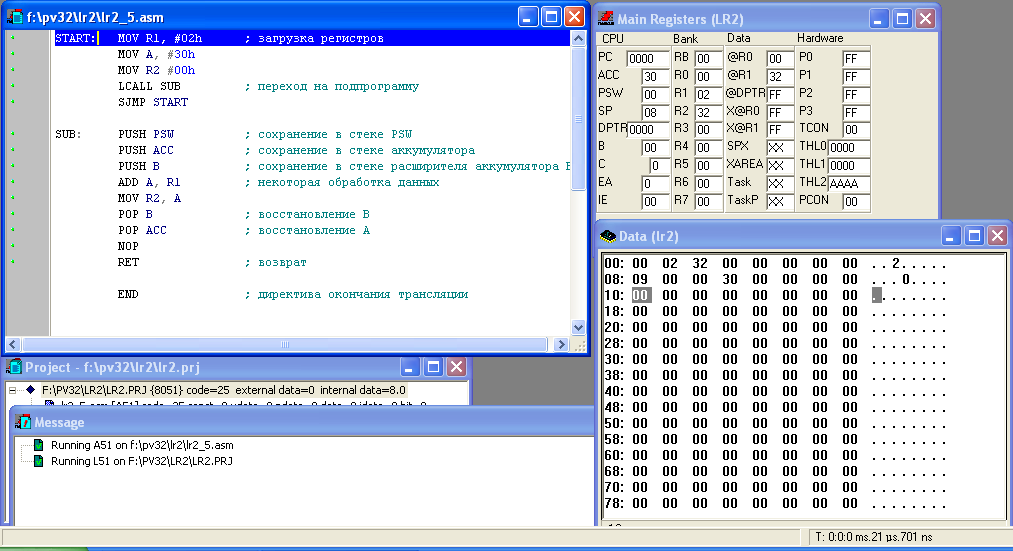
а) до замены команды POP PSW на NOP





б) после замены команды POP PSW на NOP





Т.к. команда POP PSW не была выполнена, и значение указателя SP не изменилось, то возврат был осуществлен не на адрес 0009H, а на 0000h.

**Исследование команд логических операций**

Для примера исследуем команду логического умножения регистра R1 на содержимое ячейки памяти с адресом 40h, результат помещается в аккумулятор:

**START: JMP LOG ; переход к программе**

**ORG 30H ; директива размещения программы с адр. 30h**

**LOG: MOV R1, #57h ; загрузка данных в регистры и ячейки**

**MOV 40h, #4Ch**

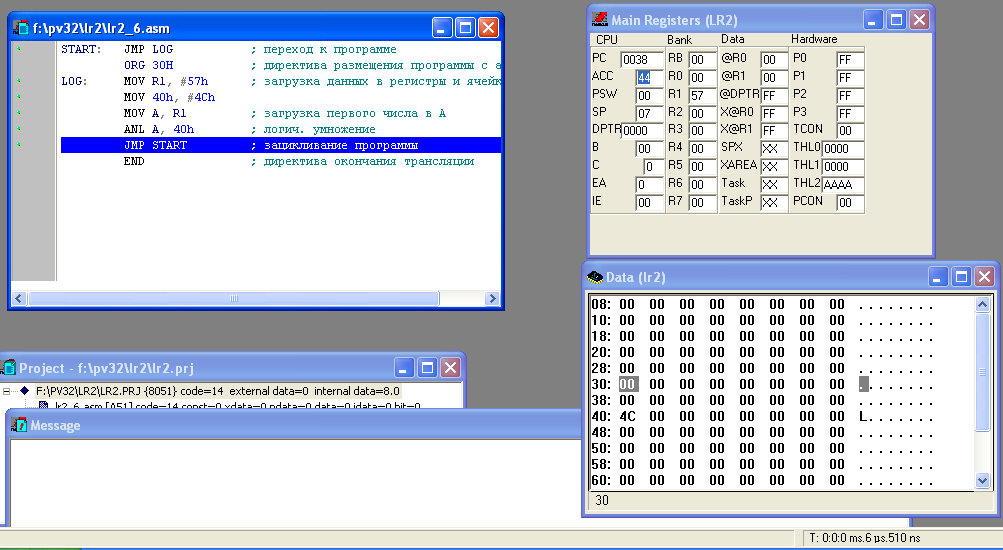
**MOV A, R1 ; загрузка первого числа в A**

**ANL A, 40h ; логич. умножение**

**JMP START ; зацикливание программы**

**END ; директива окончания трансляции**

Результат выполнения программы:



**Выводы**

В ходе лабораторной работы были достигнуты поставленные цели: изучена система команд микроконтроллеров семейства *Intel 8051*, а также закреплены навыки работы с интегрированной средой ProView фирмы Franklin Software Inc.

Были проведены исследования заданных примеров, исправлена программа умножения, определены времена выполнения программ, в программах сложения и умножения время выполнения на прямую зависит от формата чисел.

При исследовании команд работы со стеком было определено что при замене команды происходит сбой при возврате из подпрограммы вследствие не верного значения указателя вершины стека.

Так же в отчете представлен пример исследования логических операций, вынесенных на самостоятельное изучение.