**Лабораторная работа 5**

**Программы с использованием стека подпрограмм**

**Цель работы:** изучить команды вызова подпрограмм и возврата из них, команды работы со стеком. Научиться писать и выполнять программы с использованием подпрограмм.

В данной лабораторной работе была написана подпрограмма, инициализирующую область памяти длины len слов начиная с адреса adr значением pat. А также создал макрос с функциями, аналогичными подпрограмме, написанной ранее. Начальный адрес памяти adr, длину len и данные pat передал макросу в качестве параметров. Написанный код:

**init**

;Пример инициализации TMS320C40

;Создание таблицы векторов прерываний

.include "subr.asm"

\_myvect .sect "myvect" ;Создаём секцию для векторов прерываний

reset .word \_c\_int00 ;RESET вектор

.space 1 ;Резервируем место для NMI прерывания

;Создаём таблицу векторов программных прерывани

\_mytrap .sect"mytrap" ;Секция для векторов программных прерываний

;Создаём стек

\_mystack .usect"mystack",100 ;резервирусем 100 слова под стек

.text

stacka .word \_mystack ;адрес mystack секции

ivta .word \_myvect ;адрес myvect секции

tvta .word \_mytrap ;адрес mytrap секции

ieval .word 1 ;значения регистра разрешения прерываний

gctrl .word 1EE78000h ;значения регистра управления памятью

lctrl .word 1EF78000h ;значения регистра управления памятью

mctrla .word 100000h ;адрес регистра управления глобальной памятью

\_c\_int00: ;точка входа в программу, при reset начинаем отсюда

;Инициализация регистра DP

ldp stacka

;Устанавливаем указатель на таблицу прерываний

ldi @ivta,AR0

ldpe AR0,IVTP

;Устанавливаем указатель на таблицу программных прерываний

ldi @tvta,AR0

ldpe AR0,TVTP

;Инициализируем регистр управления глобальной памятью

ldi @mctrla,AR0

ldi @gctrl,R0

sti R0,\*AR0

;Инициализируем регистр управления локальной памятью

ldi @lctrl,R0

sti R0,\*+AR0(4)

;Инициализируем указатель стека

ldi @stacka,SP

;Инициализируем регистр разрешения прерываний

;Запись 1 в IIE разрешает прерывание таймера

ldi @ieval,IIE

;Глобально разрешаем прерывания, очищаем и разрешаем кэш

or 3800h,ST

BR begin ;переход к началу приложения

nop

nop

nop

begin ;начало программы

call InitAR

stop br stop ;конец программы

nop

nop

nop

.end

**Prog**

InitProg

len .word 2h

adr .word 80001000h

pat .word 0FFFFFFFh

ldi @adr, AR0

ldi @len, AR1

ldi @pat, AR2

ldi AR1, RC

rptb loop

sti AR2, \*AR0

addi 1, AR0

loop nop

ldi 0, AR0

ldi 0, AR1

ldi 0, AR2

rets

nop

nop

nop

**subr**

InitAR

ldi 0, AR0

ldi 0, AR1

ldi 0, AR2

ldi 0, AR3

ldi 0, AR4

ldi 0, AR5

ldi 0, AR6

ldi 0, AR7

rets

nop

nop

nop

Ниже приведены результаты написанных программ (Рисунок1, Рисунок 2, Рисунок3).

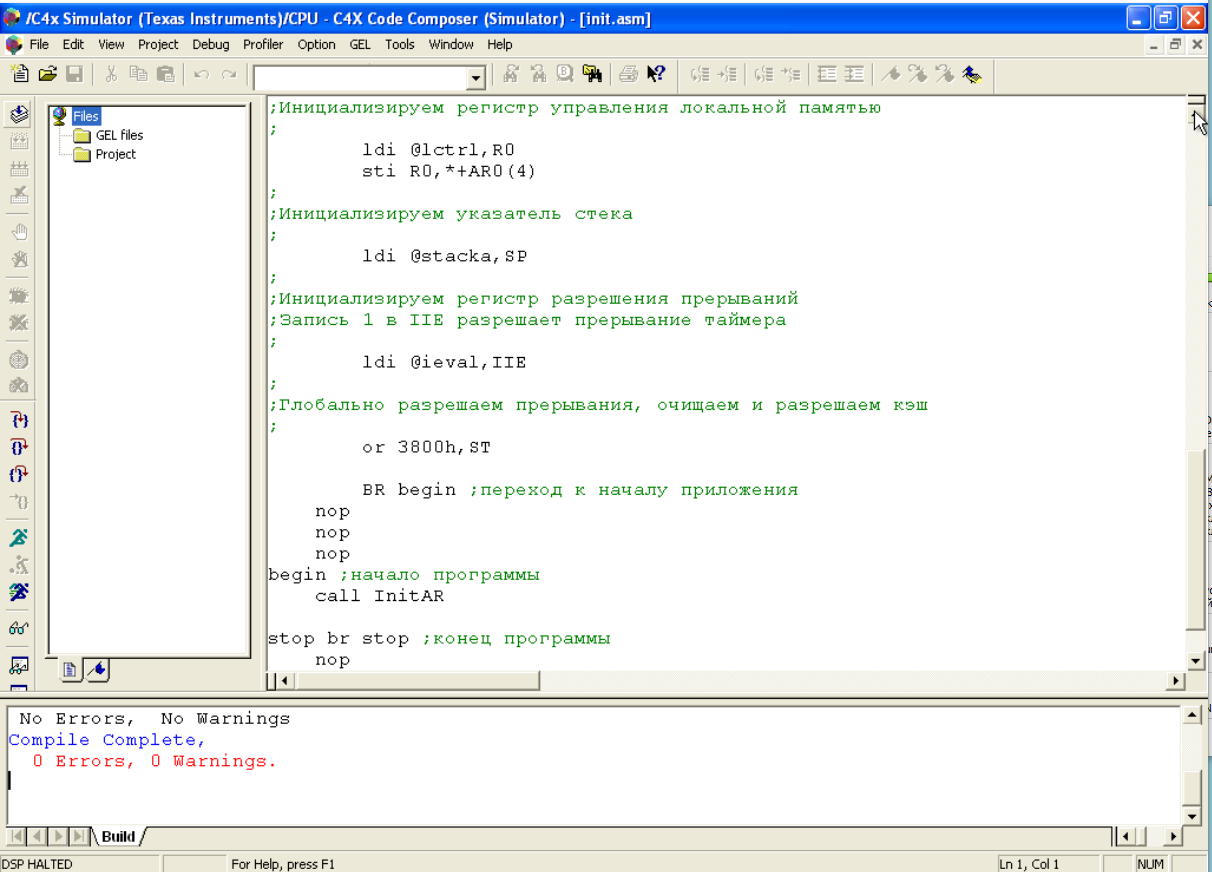


Рисунок 1 – Результат компиляции

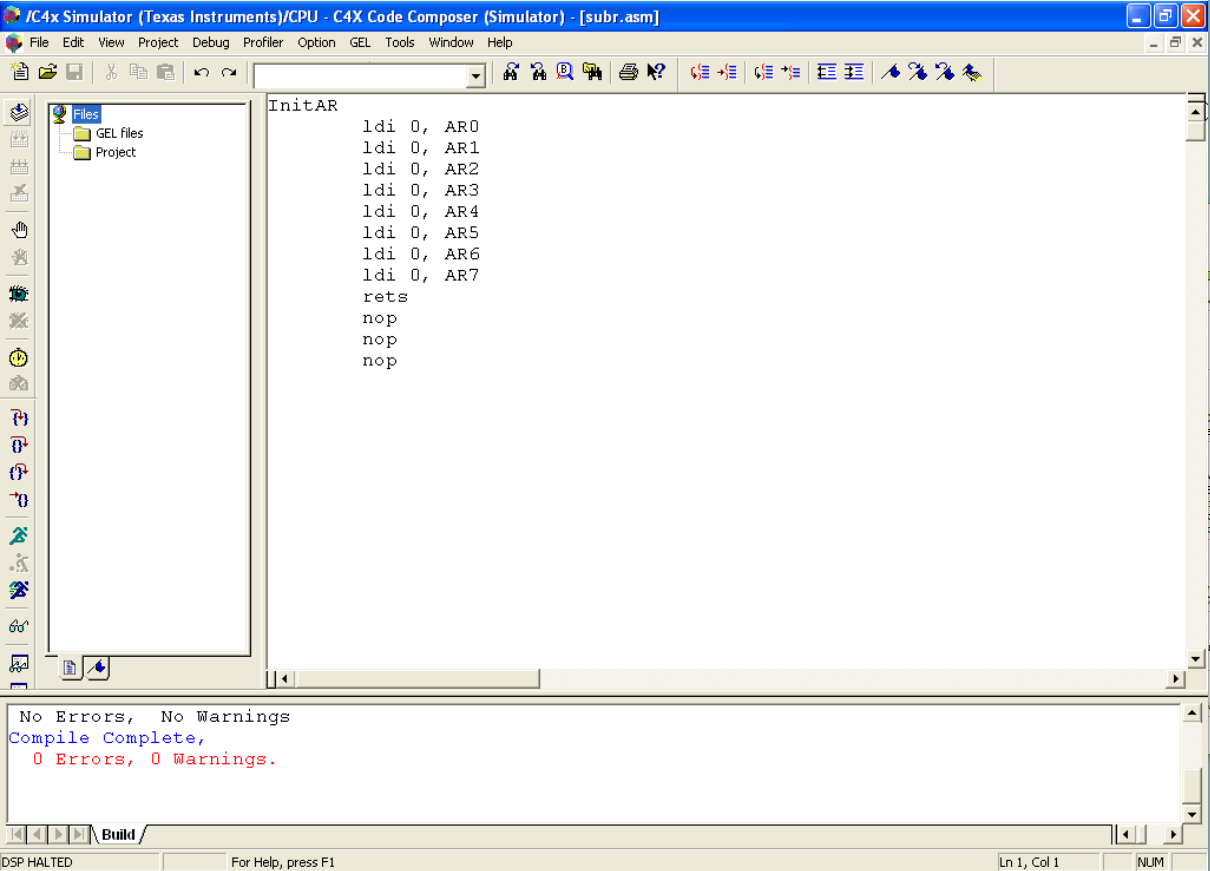


Рисунок 2 – Результат компиляции

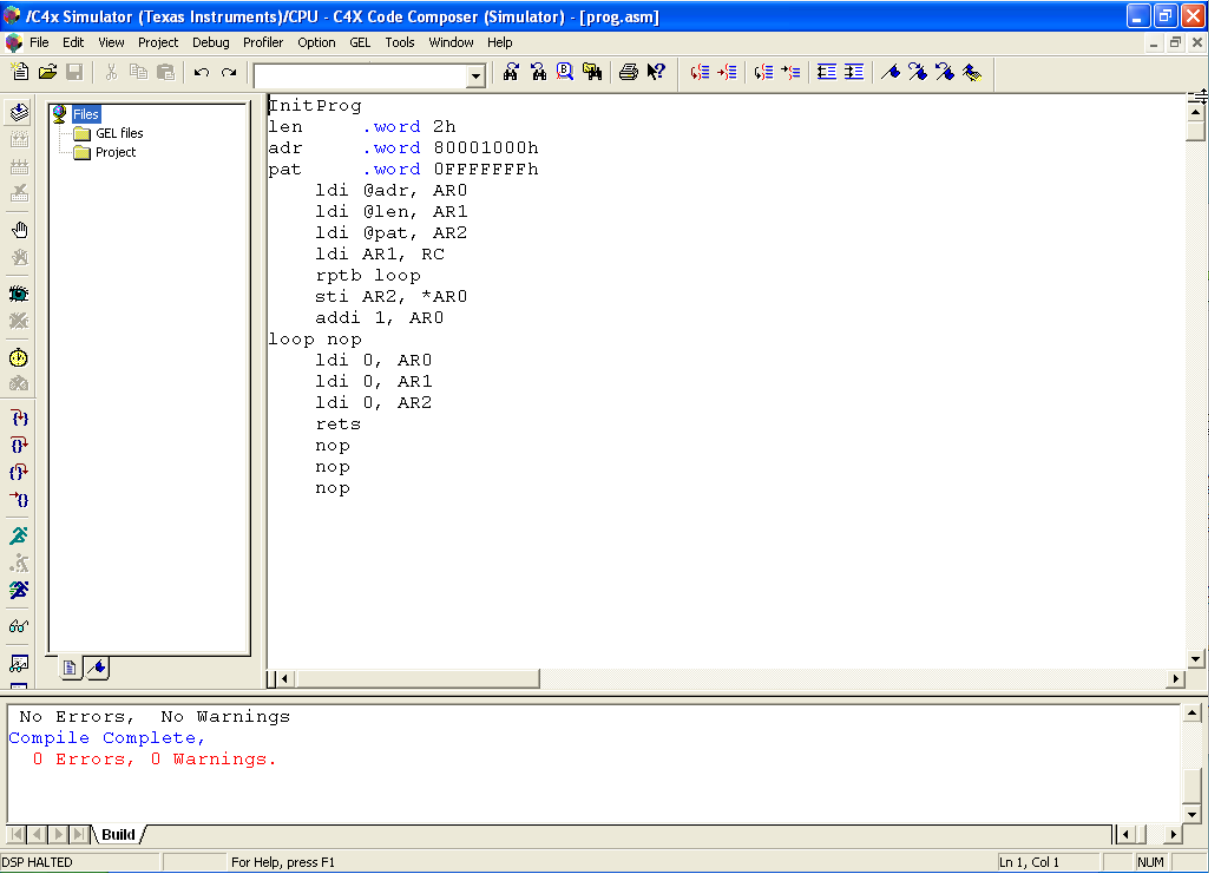


Рисунок 3 – Результат компиляции