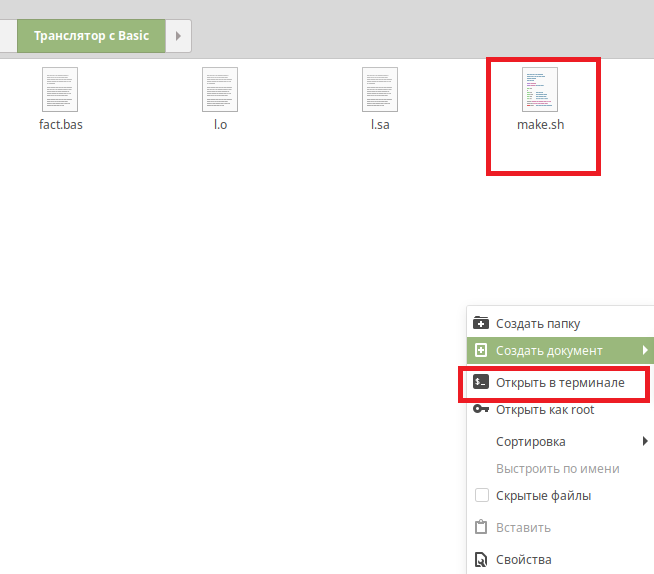
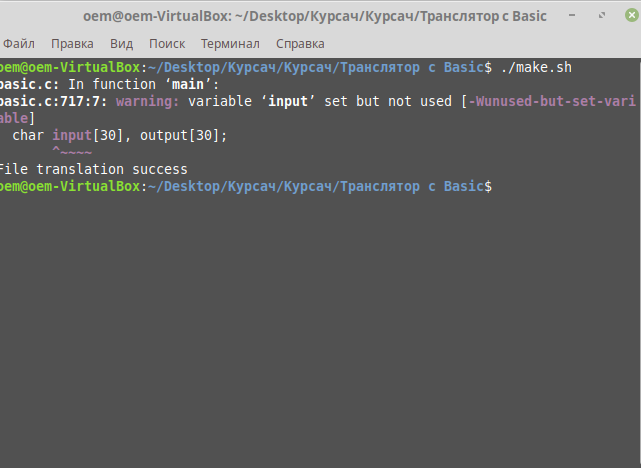
Для корректной компиляции необходимо выполнить следующую команду с root правами:  
sudo apt-get install libc6-dev

На чистом минте нету этого модуля, поэтому данное действие должно разрешить проблему компиляции.

Условно, курсовая разделена на три модуля:

1. Machine - Программная панель, включающая в себя работу вычислительной машины Simple Computer
2. Basic translator - Транслятор с высокоуровневого языка Simple Basic в язык Simple Assembler
3. Assembler translator - Транслятор с языка Simple Assembler в формат, удобный для вычислительной машины Simple Computer

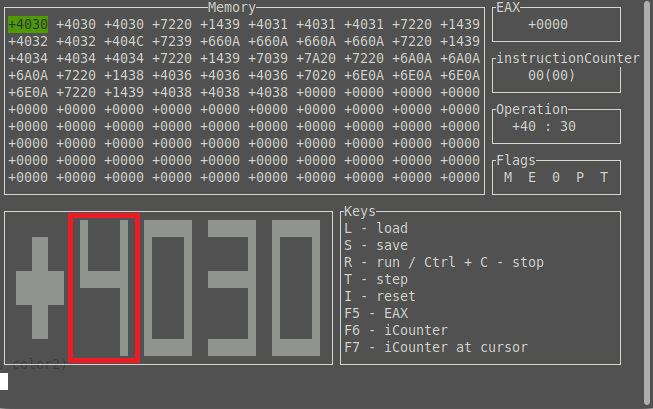
Каждый модуль компилируется с помощью находящегося там скрипта make.sh  




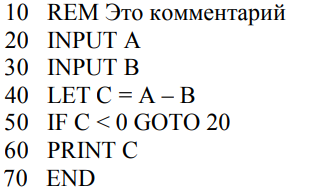
В случае, если в консоли написано: «отказано в доступе», необходимо к нужному файлу выполнить команду chmod -R 755 make.sh

1 модуль – это компиляция лабораторных работ, объединенных в курсовой проект, где реализована вычислительная машина Simple Computer. Состоит из следующих модулей:

* Main.c – главный модуль, который выполняет отрисовку/работу машины с помощью средств других модулей. Содержит в себе главные функции:
  + int ALU (int command, int operand) – реализует алгоритм работы арифметикологического устройства.
  + int CU (void) – обеспечивает работу устройства управления.
* myTerm.c – модуль работы с консолью (движение курсора по терминалу, написание текста, смена цвета букв/фона букв)
* mySimpleComputer.c – модуль, содержащий методы работы с памятью, кодированием/декодированием команд, работа с регистрами
* myReadKey.c – модуль, позволяющий обрабатывать нажатия клавиш во время работы программы машины.
* myBigChars.c – модуль для рисования «больших» символов (включая рамки):



2 модуль – транслятор с языка Simple Basic в SimpleAssembler



Некоторые примитивные команды интерпретируются примерно так:  
INPUT - > READ

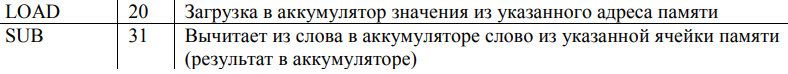
OUPUT - > WRITE

Но есть такие конструкции как if, которые так просто не получится интерпретировать. Главная идея подобной интерпретации – команды JNEG и JZ



К примеру, есть такая конструкция:  
**if** a < b **goto** **20**

Главная идея – использовать команду JNEG. Нужно привести запись a < b к виду a – b < 0, которая проще реализуется с помощью JNEG. Для этого нужны команды:



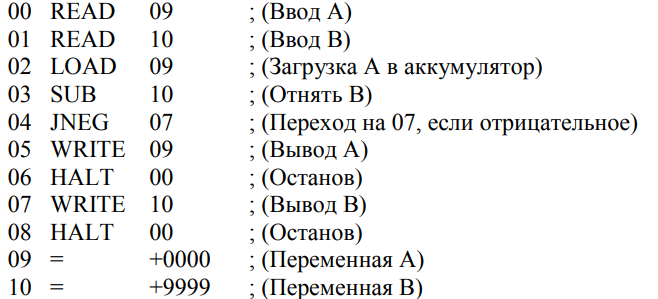
То есть, с помощью LOAD заносится значение переменной a в аккумулятор, затем используется SUB, чтобы из этого значение вычесть b, и затем применяется JNEG, т.к. в аккумуляторе к этому времени содержится результат a – b.

В случае обратной операции сравнения,  
**if** a > b **goto** **20**

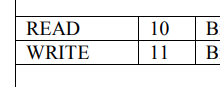
в аккумулятор сначала будет внесено значение переменной b, затем из него вычитается значение переменной a.

Подобные манипуляции используются и для остальных более сложных конструкций.

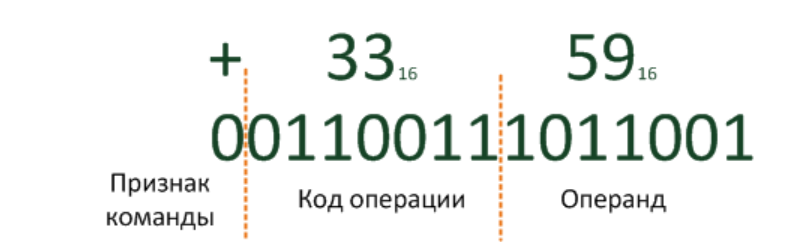
3 модуль – транслятор с языка Simple Assembler в бинарный файл, содержащий кодированные команды для машины Simple Computer



В соответствии с каждой командой (типа READ|LOAD|SUB) ставится номер команды в 16 системе счисления



Затем эти два значения кодируются в следующий вид



Это число заносится в массив, который затем будет записан в бинарный файл.

--------------------------------------------------------------------------------------------------------------------------------------

В общем случае, пользователь пишет программу на Simple Basic, с помощью транслятора с Simple Basic и файла make.sh создает файл program.sa, переносит этот файл в папку транслятора с Simple Assembler, в итоге получая файл program.binary, который переносит в папку с терминалом. Там, с помощью горячей клавиши L, вводит название файла, и он загружается в командную панель, после чего можно начать выполнение с помощью клавиши R.