Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(ФГБОУ ВО «СибГУТИ»)

*Факультет информатики и вычислительной техники*

*Кафедра вычислительных систем*

Курсовая работа по дисциплине

«Архитектура ЭВМ»

Выполнил:

студент гр. ИП-513

Майоров С.А.

Проверил:

доцент кафедры ВС

Майданов Ю.С.

Новосибирск

2017

Оглавление

[Задание к курсовой работе 3](#_Toc423377134)

[Исходный код программы 7](#_Toc423377135)

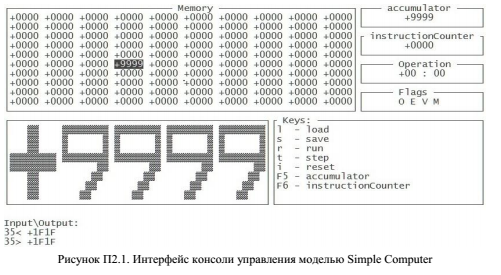
[Результаты работы программы 10](#_Toc423377136)

[Вывод. 11](#_Toc423377137)

[Список литературы 12](#_Toc423377138)

# Задание к курсовой работе

В рамках выполнения курсового проектирования необходимо разработать программную модель простейшей вычислительной машины Simple Computer. Архитектура Simple Computer представлена ниже.



Для управления моделью (определения начальных состояний узлов Simple Computer, запуска программ на выполнение, отражения хода выполнения программ) требуется создать консоль (рис. П2.1). Необходимо реализовать

трансляторы с языков Simple Assembler и Simple Basic для программирования Simple Computer.

Архитектура вычислительной машины Simple Computer

Архитектура Simple Computer представлена на рисунке П2.2 и включает следующие функциональные блоки:

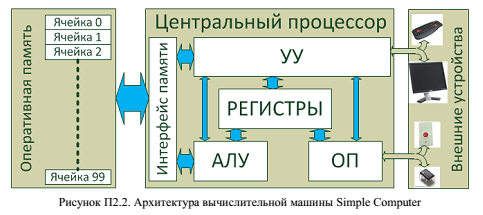
* оперативную память;
* внешние устройства;
* центральный процессор.

Таблица 1. Команды центрального процессора Simple Computer

|  |  |  |
| --- | --- | --- |
| Обозначение | Код | Значение |
| READ | 10 | Ввод с терминала в указанную ячейку памяти с контролем переполнения |
| WRITE | 11 | Вывод на терминал значение указанной ячейки памяти |
| LOAD | 20 | Загрузка в аккумулятор значения из указанного адреса памяти |
| STORE | 21 | Выгружает значение из аккумулятора по указанному адресу памяти |
| ADD | 30 | Выполняет сложение слова в аккумуляторе и слова из указанной ячейки памяти (результат в аккумуляторе) |
| SUB | 31 | Вычитает из слова в аккумуляторе слово из указанной ячейки памяти (результат в аккумуляторе) |
| DIVIDE | 32 | Выполняет деление слова в аккумуляторе на слово из указанной ячейки памяти (результат в аккумуляторе) |
| MUL | 33 | Вычисляет произведение слова в аккумуляторе на слово (результат в аккумуляторе) |
| JUMP | 40 | Переход к указанному адресу памяти |
| JNEG | 41 | Переход к указанному адресу памяти, если в аккумуляторе находится отрицательное число |
| JZ | 42 | Переход к указанному адресу памяти, если в аккумуляторе находится ноль |
| HALT | 43 | Остановка, выполняется при завершении работы программы |
| USRFUNC | 51+ | В зависимости от варианта |

В рамках курсовой работы необходимо доработать модель Simple Computer так, чтобы она обрабатывала команды, записанные в оперативной памяти. Система команд представлена в таблице 1. Из пользовательских функций необходимо реализовать только одну согласно варианту задания. Для разработки программ требуется создать трансляторы с языков Simple Assembler и Simple Basic.

Обработка команд центральным процессором

Для выполнения программ моделью Simple Computer необходимо реализовать две функции:

* int ALU (int command, int operand) – реализует алгоритм работы арифметико-логического устройства. Если при выполнении функции возникла ошибка, которая не позволяет дальше выполнять программу, то функция возвращает -1, иначе 0;
* int CU (void) – обеспечивает работу устройства управления.

Транслятор с языка Simple Assembler

Разработка программ для Simple Computer может осуществляться с использованием низкоуровневого языка Simple Assembler. Для того чтобы программа могла быть обработана Simple Computer необходимо реализовать транслятор, переводящий текст Simple Assembler в бинарный формат, которым может быть считан консолью управления. Пример программы на Simple Assembler:

00 READ 09 ; (Ввод А)

01 READ 10 ; (Ввод В)

02 LOAD 09 ; (Загрузка А в аккумулятор)

03 SUB 10 ; (Отнять В)

04 JNEG 07 ; (Переход на 07, если отрицательное)

05 WRITE 09 ; (Вывод А)

06 HALT 00 ; (Останов)

07 WRITE 10 ; (Вывод В)

08 HALT 00 ; (Останов)

09 = +0000 ; (Переменная А)

10 = +9999 ; (Переменная В)

Программа транслируется по строкам, задающим значение одной ячейки памяти. Каждая строка состоит как минимум из трех полей: адрес ячейки памяти, команда (символьное обозначе-ние), операнд. Четвертым полем может быть указан комментарий, который обязательно должен начинаться с символа точка с запятой. Название команд представлено в таблице 1. Дополнительно используется команда =, которая явно задает значение ячейки памяти в формате вывода его на экран консоли (+XXXX).

Команда запуска транслятора должна иметь вид: sat файл.sa файл.o, где файл.sa – имя файла, в котором содержится программа на Simple Assembler, файл.o – результат трансляции.

Транслятор с языка Simple Basic

Для упрощения программирования пользователю модели Simple Computer должен быть предоставлен транслятор с высокоуровневого языка Simple Basic. Файл, содержащий программу на Simple Basic, преобразуется в файл с кодом Simple Assembler. Затем Simple Assembler-файл транслируется в бинарный формат.

В языке Simple Basic используются следующие операторы: rem, input, output, goto, if, let, end. Пример программы на Simple Basic:

10 REM Это комментарий

20 INPUT A

30 INPUT B

40 LET C = A – B

50 IF C < 0 GOTO 20

60 PRINT C

70 END

Каждая строка программы состоит из номера строки, оператора Simple Basic и параметров. Номера строк должны следовать в возрастающем порядке. Все команды за исключением команды конца программы могут встречаться в программе многократно. Simple Basic должен оперировать с целыми выражениями, включающими операции +, -, \*, и /. Приоритет операций аналогичен C. Для того чтобы изменить порядок вычисления, можно использовать скобки.

Транслятор должен распознавания только букв верхнего регистра, то есть все символы в программе на Simple Basic должны быть набраны в верхнем регистре (символ нижнего регистра приведет к ошибке). Имя переменной может состоять только из одной буквы. Simple Basic оперирует только с целыми значениями переменных, в нем отсутствует объявление переменных, а упоминание переменной автоматически вызывает её объявление и присваивает ей нулевое значение. Синтаксис языка не позволяет выполнять операций со строками

# Исходный код программы

#include "SimpleComputer.h"

int main(int argc, char \*\*argv)

{

signal (SIGALRM, IncInstCount);

signal (SIGUSR1, Stop);

int i, j;

InstCount = 0;

int value = 0, flag;

int com, oper;

char mem[6];

int n;

initialize();

//MECHANISM

int prev = InstCount;

int prevx, prevy, InstCountx, InstCounty;

int flagstep = 1;

int exit = 0;

enum keys button;

while(!exit)

{

print\_value();

mt\_gotoXY(5, 70);

n = sprintf(mem, "%04d", InstCount);

write(STDOUT\_FILENO, mem, n);

print\_Accum();

big\_window();

print\_flag();

operation();

rk\_mytermregime(1, 0, 1, 0, 1);

mt\_gotoXY(23, 7);

rk\_readkey(&button);

sc\_regGet(REG\_STEP\_IGNORE, &flagstep);

if(!flagstep)

{

switch(button)

{

case r\_key:

raise(SIGUSR1);

break;

case i\_key:

refresh();

break;

default:

break;

}

continue;

}

switch(button)

{

case l\_key:

sc\_memoryLoad("satexample.o");

for(i = 0; i < 10; ++i)

for(j = 0; j < 10; ++j)

write\_ram(i, j);

break;

case s\_key:

sc\_memorySave("satexample.o");

break;

case r\_key:

alarm(1);

sc\_regInit();

break;

case t\_key:

CU();

break;

case i\_key:

refresh();

break;

case f5\_key:

set\_Accum();

break;

case f6\_key:

set\_InstCount();

break;

case enter\_key:

enter\_ram();

break;

case up\_key:

if((InstCount - 10) >= 0)

InstCount -= 10;

break;

case down\_key:

if((InstCount + 10) <= 99)

InstCount += 10;

break;

case left\_key:

if((InstCount - 1) >= 0)

InstCount--;

break;

case right\_key:

if((InstCount + 1) <= 99)

InstCount++;

break;

case quit\_key:

exit = 1;

break;

default:

break;

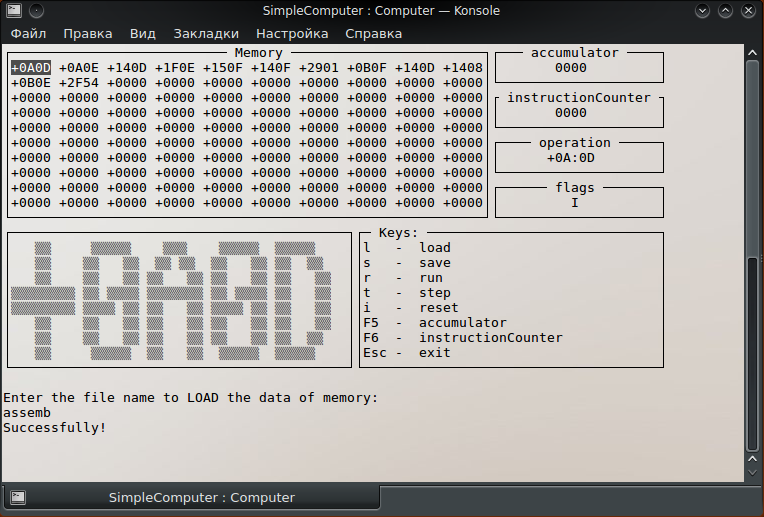
}

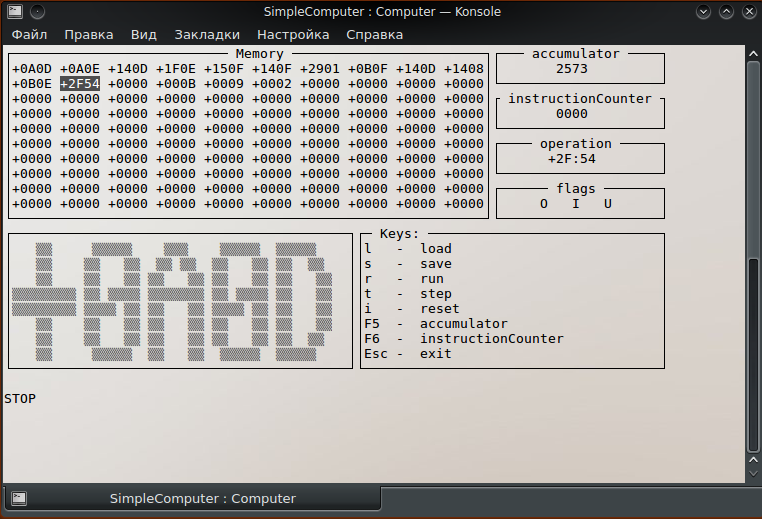
}

return 0;

}

# Результаты работы программы





# Вывод

Simple Computer - модель простейшей ВМ, исполняющая машинные команды, заданные явно или сформированные путем трансляции с языков программирования Simple Assembler и Simple Basic, в интерактивном режиме. Пользователь имеет возможность полностью контролировать процесс выполнения и отслеживать ошибки.

Виртуальную машину составляют блоки, реализующие аналогичный функционал по сравнению с соответственными блоками реальной ЭВМ.

Это оперативная память, хранящая программы и данные, регистры, АЛУ и УУ, обработчик прерываний от внешних устройств, клавиатура, монитор, системный таймер, кнопка “Reset”.

Модель обладает своей системой команд и трансляторами, (явно или неявно) включающими в себя лексический, синтаксический анализатор и генератор кода.

Разработка Simple Computer позволила получить обширные сведения в области организации реальных вычислительных машин.

# Список литературы

1. Архитектура компьютера. Э. Таненбаум. 5-е изд. - СПб.: 2007. — 844 с.
2. Язык программирования Си. Керниган Б., Ритчи Д. 3-е изд., испр. —СПб.: "Невский Диалект", 2001. - 352 с
3. Мамойленко С.Н., Молдованова О.В. ЭВМ и периферийные устройства: Учебное пособие. – Новосибирск: СибГУТИ, 2012. – 106 с.