Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
Кафедра Вычислительных систем (ВС)

ОТЧЕТ
о практической работе
«Моделирование архитектуры фон Неймана на языке С.»
Вариант №2

Работу выполнил:
студент 1 курса
группы ИС-541
Устюжанин Д.К.

г. Новосибирск
2025 г.

Содержание

Цель работы

Разработать программную модель процессора, демонстрирующую ключевые принципы архитектуры фон Неймана: единую память для команд и данных, последовательное выполнение инструкций и использование счётчика команд, через создание эмулятора на языке C с системой базовых команд и операций переходов.

Описание программы

В программе реализован уровень «5».

Виртуальная машина поддерживает следующие команды:

0. STOP – завершение программы.
1. LOAD – загрузить число хранимое в регистре.
2. ADD – прибавить к хранимому в регистре числу.
3. PRINT – вывести значение, хранимое в регистре.
4. JUMP – безусловный переход.
5. JUMP IF ZERO – условный переход.

Модель использует массив как единую память для хранения команд и данных, переменную-счётчик для отслеживания текущей команды и «регистр» для промежуточных результатов. Особенностью реализации являются безусловные и условные переходы с проверкой границ памяти.

Пример выполнения программы:

Входной массив: {1, 5, 2, -1, 3, 5, 10, 4, 2, 3, 0}.

Выполнение:

- pc=0: команда 1 → взять program[1]=10 → acc = 10 → pc += 2
- pc=2: команда 3 → вывод 10 → pc += 1
- pc=3: команда 2 → взять program[4]=5 → acc = 10 + 5 = 15 → pc += 2
- pc=5: команда 3 → вывод 15 → pc += 1
- pc=6: команда 5 → условие acc=15 (ложь) → переход на адрес 8 → pc += 2
- pc=8: команда 3 → вывод 15 → pc += 1
- pc=9: команда 0 → стоп

Вывод программы:

```
Выполняется команда 1, acc = 10
Выполняется команда 3, 10, acc = 10
Выполняется команда 2, acc = 15
```

```
Выполняется команда 3, 15, acc = 15
Выполняется команда 5, acc = 15
Выполняется команда 3, 15, acc = 15
Программа завершена.
```

Листинг программы

```
#include <stdio.h>

int main() {
  // хранит команды и данные
  int program[] = {1, 37, 2, 81, 4, 6, 1, 0, 3, 5, 13, 5, 6, 0};
  size_t size = sizeof program / sizeof program[0]; // размер
массива
  int pc=0; // счетчик команд, указывающий на текущую команду
  int acc=0; // регистр-аккумулятор хранит текущий результат

  while (program[pc]) { // работает пока не встретит 0 (STOP)
    printf("Выполняется команда %d, ", program[pc]);
    switch (program[pc]) {
      case 1: // LOAD – загрузить число хранимое в регистре
        acc = program[pc+1]; pc+=2;
        break;
      case 2: // ADD – прибавить к хранимому в регистре числу
        acc += program[pc+1]; pc+=2;
        break;
      case 3: // PRINT – вывести значение, хранимое в регистре
        pc++;
        printf("%d, ", acc);
        break;
      case 4: // JUMP – безусловный переход
        if (program[pc+1] >= size) {
          program[pc]=0; // stops program
```

```
            printf("переход за границы памяти! ");
          } else pc = program[pc+1];
          break;
        case 5: // JUMP IF ZERO – условный переход
          if (acc==0) {
            if (program[pc+1] < size) pc = program[pc+1];
            else {
              program[pc]=0;
              printf("переход за границы памяти! ");
            }
          } else pc += 2;
          break;
        default: // неизвестная команда
          printf("неизвестная команда! ");
          program[pc]=0;
      }
      printf("acc = %d\n", acc);
    }
    printf("Программа завершена.\n");

    return 0;
}
```

Программа полностью соответствует принципам архитектуры фон Неймана, так как в ней присутствуют единая память (массив program), последовательное выполнение (while), счетчик команд (pc) и управление потоком (команды 4 и 5).

Результат выполнения программы

```
Выполняется команда 1, acc = 37
Выполняется команда 2, acc = 118
Выполняется команда 4, acc = 118
Выполняется команда 1, acc = 0
Выполняется команда 3, 0, acc = 0
Выполняется команда 5, acc = 0
Программа завершена.
```

Объяснение работы условного (JUMP IF ZERO) и безусловного (JUMP) переходов в текущей программе:

Безусловный переход (команда 4) всегда изменяет счётчик команд на значение из следующей ячейки памяти: `pc = program[pc+1]`. Условный переход (команда 5) проверяет состояние аккумулятора: при `acc == 0` выполняется переход по указанному адресу, иначе счётчик увеличивается на 2 для пропуска операнда. Оба перехода включают проверку границ памяти: при попытке перехода за пределы массива программа аварийно завершается установкой команды STOP.

Объяснение работы условного (JUMP IF ZERO) и безусловного (JUMP) переходов на примере:

Входной массив: {1, 3, 3, 2, -1, 5, 9, 4, 2, 0}.

Выполнение:

- pc=0: команда 1 → взять program[1]=3 → acc = 0 + 3 = 3 → pc += 2
- pc=2: команда 3 → вывод значения acc=3 → pc += 1
- pc=3: команда 2 → взять program[4]=-1 → acc = 3 + (-1) = 2 → pc += 2
- pc=5: команда 5 → acc=2!=0 → pc += 2
- pc=7: команда 4 → взять program[8]=2 → pc = 2
- pc=2: команда 3 → вывод значения acc=2 → pc += 1
- pc=3: команда 2 → взять program[4]=-1 → acc = 2 + (-1) = 1 → pc += 2

- pc=5: команда 5 → acc=1!=0 → pc += 2
- pc=7: команда 4 → взять program[8]=2 → pc = 2
- pc=2: команда 3 → вывод значения acc=1 → pc += 1
- pc=3: команда 2 → взять program[4]=-1 → acc = 1 + (-1) = 0 → pc += 2
- pc=5: команда 5 → acc==0 → взять program[6]=9 → pc = 9
- pc=9: команда 0 → стоп

Вывод программы:

```
Выполняется команда 1, acc = 3
Выполняется команда 3, 3, acc = 3
Выполняется команда 2, acc = 2
Выполняется команда 5, acc = 2
Выполняется команда 4, acc = 2
Выполняется команда 3, 2, acc = 2
Выполняется команда 2, acc = 1
Выполняется команда 5, acc = 1
Выполняется команда 4, acc = 1
Выполняется команда 3, 1, acc = 1
Выполняется команда 2, acc = 0
Выполняется команда 5, acc = 0
Программа завершена.
```

Программа реализует обратный отсчёт от 3 до 1. Безусловный переход (команда 4) циклически возвращает выполнение к команде PRINT, создавая повторяющуюся последовательность. Условный переход (команда 5) проверяет достижение нуля - при acc=0 программа завершается, иначе продолжает цикл. Это демонстрирует организацию ветвления и циклического выполнения через комбинацию условных и безусловных переходов.

Вывод

В ходе работы успешно реализована модель процессора, демонстрирующая принципы архитектуры фон Неймана. Удалось создать систему с единой памятью для команд и данных, последовательным выполнением и управлением потоком через переходы. Основные трудности связаны с защитой от ошибок памяти, которые решены через проверки границ и преобразование ошибочных команд в STOP. Модель подтвердила понимание фундаментальных основ организации вычислительных систем.