**Тверской Государственный технический университет**

Кафедра: Программного обеспечения и вычислительной техники

Отчёт по лабораторной работе № 2

Выполнил: Студент первого курса. Группы Б.ПИН.РИС 18.06 ФИО Студента

Проверил: Калабин Александр Леонидович

**Тверь 2018г.**

**План работы**

**1.Изучите команды Ассемблера, входящие в программы 1 и 2**

**2. Изучите, прокомментируйте и сравните две программы**

**SEG1 SEGMENT**

**OPER1 DB 5**

**OPER2 DB 3**

**SEG1 ENDS**

**SEG2 SEGMENT**

**RESULT DB ?**

**SEG2 ENDS**

**CODE SEGMENT**

**ASSUME CS:CODE, DS:SEG1**

**START: MOV AX,SEG1**

**MOV DS,AX**

**MOV AH,OPER1**

**ADD AH,OPER2**

**ASSUME DS:NOTHING, DS:SEG2**

**MOV AX,SEG2**

**MOV DS,AX**

**MOV RESULT,AH**

**MOV AH,4CH**

**INT 21H**

**CODE ENDS**

**END START**

**Программа 2**

**SEG1 SEGMENT**

**OPER1 DB 5**

**OPER2 DB 3**

**SEG1 ENDS**

**SEG2 SEGMENT**

**RESULT DB ?**

**SEG2 ENDS**

**CODE SEGMENT**

**ASSUME CS: CODE, DS:SEG1, ES:SEG2**

**START: MOV AX, SEG1**

**MOV DS, AX**

**MOV AX, SEG2**

**MOV ES, AX**

**MOV BH, OPER1**

25

**ADD BH, OPER2**

**MOV RESULT, BH**

**MOV AH, 4CH**

**INT 21H**

**CODE ENDS**

**END START**

**3.Выберите более эффективную программу и получите загрузочный модуль в соответствии с пунктами 1 и 2.**

**4. Сформулируйте критерии оценки эффективности программ.**

**Выполнение поставленных задач**

**2) SEG1 SEGMENT // Объявление первого сегмента с именем SEG1**

**OPER1 DB 5 // Объявление первого операнта с именем OPER1 с типом байт и значением 5**

**OPER2 DB 3 // Объявление первого операнта с именем OPER2 с типом байт и значением 3**

**SEG1 ENDS //Завершение первого сегмента**

**SEG2 SEGMENT // Объявление первого сегмента с именем SEG2**

**RESULT DB ? // Объявление первого операнта с именем RESULT с типом байт и значением ?**

**SEG2 ENDS //Завершение второго сегмента**

**CODE SEGMENT //Вход в составляющую кода сегмента с именем CODE**

**ASSUME CS:CODE, DS:SEG1 // Директиву ASSUME CS: требуется указывать в каждой программе, в которой используются стандартные сегментные директивы, так как Ассемблеру необходимо знать о сегменте** **Кода для того, чтобы установить выполняемую программу. Кроме того, обычно используются директива ASSUME DS благодаря ,которой Ассемблер знает, к каким ячейкам памяти мы можем адресоваться в данный момент. В этом случае мы заносим в сегмент счётчика сегмент кода с названием код и в сегмент данных данные сегмента 1**

**START: MOV AX,SEG1 // Запускаем начало выполнения программы и заносим в регистр аккумулятора данные из первого сегмента.**

**MOV DS,AX // Переносим данные из регистра аккумулятора в сегмент данных**

**MOV AH,OPER1 // Заносим оперант1 со значением 5 в регистр высшего порядка**

**ADD AH,OPER2 // Складываем значение занесенное в регистр аккумуляторного высшего порядка со заносящим в него значением операнта2(значение 3)**

**ASSUME DS:NOTHING, DS:SEG2 // Директиву ASSUME CS: требуется указывать в каждой программе, в которой используются стандартные сегментные директивы, так как Ассемблеру необходимо знать о сегменте** **Кода для того, чтобы установить выполняемую программу. Кроме того, обычно используются директива ASSUME DS благодаря ,которой Ассемблер знает, к каким ячейкам памяти мы можем адресоваться в данный момент. В этом случае мы заносим в сегмент данных значение ничего ,а в сегмент данных данне сегмента 2**

**MOV AX,SEG2 // Запускаем начало выполнения программы и заносим в регистр аккумулятора данные из второго сегмента.**

**MOV DS,AX // Переносим данные из регистра аккумулятора в сегмент данных**

**MOV RESULT,AH // Переносим значение полученного в аккумуляторном регистре высшего порядка в ячейку RESULT**

**MOV AH,4CH**

**// Сложные команды преобразования полученного результата из машинного вида в понятный человеку вид**

**INT 21H**

**CODE ENDS // Выход из составляющей кода сегмента с именем CODE**

**END START // Завершаем выполнение программы**

**// Объявление первого сегмента с именем SEG1**

**OPER1 DB 5 // Объявление первого операнта с именем OPER1 с типом байт и значением 5**

**OPER2 DB 3 // Объявление первого операнта с именем OPER2 с типом байт и значением 3**

**SEG1 ENDS //Завершение первого сегмента**

**SEG2 SEGMENT // Объявление второго сегмента с именем SEG2**

**RESULT DB ? // Объявление первого операнта с именем RESULT с типом байт и значением ?**

**SEG2 ENDS //Завершение второго сегмента**

**CODE SEGMENT //Вход в составляющую кода сегмента с именем CODE**

**ASSUME CS: CODE, DS:SEG1, ES:SEG2 // Директиву ASSUME CS: требуется указывать в каждой программе, в которой используются стандартные сегментные директивы, так как Ассемблеру необходимо знать о сегменте** **Кода для того, чтобы установить выполняемую программу. Кроме того, обычно используются директива ASSUME DS благодаря ,которой Ассемблер знает, к каким ячейкам памяти мы можем адресоваться в данный момент. В этом случае мы заносим в числовой сегмент данные из составляющей кода сегмента с именем СОDE, в сегмент данных данные сегмента 1 ,а также в сегмент дополнительных данных мы вносим данные сегмента 2**

**START: MOV AX, SEG1 // Запускаем начало выполнения программы и заносим в регистр аккумулятора данные из первого сегмента.**

**MOV DS, AX // Переносим данные из регистра аккумулятора в сегмент данных**

**MOV AX, SEG2 // Заносим в регистр аккумулятора данные из второго сегмента.**

**MOV ES, AX // Заносим данные из аккумуляторного регистра в сегмент дополнительных данных**

**MOV BH, OPER1 // Переносим значение операнта 1 (значение 5) в регистр базы высшего порядка**

25

**ADD BH, OPER2 // Складываем значение занесенное в регистр базы высшего порядка со заносящим в него значением операнта2(значение 3)**

**MOV RESULT, BH // Переносим значение полученного в базовом регистре высшего порядка в ячейку RESULT**

**MOV AH, 4CH**

**// Сложные команды преобразования полученного результата из машинного вида в понятный человеку вид**

**INT 21H**

**CODE ENDS // Выход из составляющей кода сегмента с именем CODE**

**END START // Завершаем выполнение программы**

**Сравнение**

**Результат выполнения обоих программ будет одинаков это сложение 5+3 = 8 ,однако первая программа будет задействовать под своё выполнение больше памяти компьютера из-за повторного использования команды ASSUME DS:NOTHING, DS:SEG2 и выполнения определённых действий. И следовательно первая программа менее эффективно будет выполняться чем вторая. Во второй программе также как и в первой используются два сегмента данных вводимых в память компьютера, эти данные, которые получаться в результате выполнения команд программы не передаются второй раз через команду ASSUME DS:NOTHING, DS:SEG2 как в первой программе в сегмент данных.**

**Вторая программа задействует дополнительный сегмент данных тем самым равномерно и грамотно использовав память компьютера.**

**Критерии оценки эффективности программ**

* **Время выполнения программы**
* **Используемый объём оперативной и внешней памяти**