МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования

«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Физико-технический факультет

Кафедра оптоэлектроники

Сборник задач по основам алгоритмизации на Ассемблере для обучающихся бакалавриата по направлениям подготовки 11.03.02 «Инфокоммуникационные технологии и системы связи», 11.03.01 «Радиотехника», 11.03.04 «Электроника и наноэлектроника»

Составитель: преподаватель Гусев АА.

Краснодар 2017

Представленные в настоящем сборнике задачи направлены на формирование у обучающихся навыков составления элементарных алгоритмов с использованием ограниченного множества команд эмулятора 8-битного Ассемблера.

Исполнение задач возможно в учебном эмуляторе 8-битного Ассемблера https://schweigi.github.io/assembler-simulator/

Справочник по системе команд эмулятора - https://schweigi.github.io/assembler-simulator/instruction-set.html

Оглавление

Задача 1

В среде эмулятора 8-битного ассемблера написать программу для расчета числа действительных корней квадратного уравнения.
Вывести на output 3 варианта, в зависимости от значений коффициентов a, b, c: "2 корня", "1 корень", "нет корней"
Ниже приведен код примерной программы, оценивающей значение выражения $x1^2+2x^3-3x^2+4x^4$:

```
JMP start
str1: DB "Vyr>0"
      DB 0
str2: DB "Vyr=0"
      DB 0
str3: DB "Vyr<0"
      DB 0
x1: DB 1
x2: DB 9
x3: DB 4
x4: DB 2
start:
  MOV A, [x1]
  MUL A
  MOV B, A ; B=x1^2
   MOV A, [x3]
   MUL 2
   MOV C, A ; C=2x3
   MOV A, [x4]
   MUL 4
   MOV D, A ; D=4x4
   MOV A, [x2]
   MUL 3 ; A=3x2
   ADD B, C ; x1^2+2x3
   ADD B, D
   SUB B, A
   JC ishod3
   CMP B, 0
   JA ishod1
   JE ishod2
ishod1:
 MOV C, str1
 MOV D, 232
 CALL print
 HLT
ishod2:
 MOV C, str2
 MOV D, 232
 CALL print
```

```
 HLT
ishod3:
 MOV C, str3
 MOV D, 232
 CALL print
 HLT
print:
 PUSH A
 PUSH B
 MOV B, 0
.loop:
 MOV A, [C]
 MOV [D], A
 INC C
 INC D
 CMP B, [C]
 JNZ .loop
 POP B
 POP A
 RET
```

Задача 2

Используя код, разработанный вами в предыдущей работе, напишите программу, рассчитывающую значение дискриминанта квадратного уравнения и выводящую это значение на output.
При разработке программы учтите, что output выводит исключительно по одному символу, поступающему в ячейки памяти с номерами 232, 233 и т.д.
Чтобы вывести, например, число 23 нужно передать на output (команда MOV [D], A) два символа: '2' и '3'.
Согласно таблице ASCII код символа '2' это 50, '3' - 51. Таким образом, для вывода числового содержимого регистров на output, необходимо прибавлять число 48 (код символа '0') к цифре, которую вы хотите напечатать.
Для получения разрядов десятичного числа используйте целочисленное деление на 10.
Например 23 div 10 = 2. Таким образом вы получаете старший разряд двоичного числа, прибавляете 48 и можно отправить на output.
Для получения младшего разряда необходимо умножить старший разряд на 10 и вычесть из исходного числа: 23-2x10=3. Для вывода разряда на output вы так же прибавляете 48.

Код функции print для осуществления этих операций может иметь примерно следующий вид:

```
print:
 PUSH A
 PUSH B
 ; команды для расчета старшего разряда
; старший разряд в регистре A
ADD A, 48
 MOV [D], A
 INC D
; команды для расчета младшего разряда
ADD A, 48
MOV [D], A
 POP B
 POP A
 RET
```

Для вывода трехзначных чисел необходимо модифицировать функцию print для вычисления старшего разряда 3-значного числа (div 100).

Задача 3

В позиционных системах счисления "сдвиг" числа на N позиций влево приводит к умножению этого числа на основание системы счисления в степени N.
Например, десятичное число 45. Если сдвинуть 45 на 1 позицию влево и дописать справа 0 получится 450, т.е. 45 * 10 в 1 степени. Если сдвинуть еще на одну позицию влево и дописать справа 0, получится 4500 или, что то же самое 45 * 10 во 2 степени, т.к. 4500 получается сдвигом числа 45 на две позиции влево.
Двоичная система счисления является позиционной и при сдвиге влево происходит умножение двоичного числа на 2 в степени N, где N - количество разрядов сдвига.
Например, 11 (3) - > 110 (6) -> 1100 (12) -> 11000 (24).
Для сдвига числа на N позиций влево в Ассемблере существует команда SHL (shift left). Формат команды SHL reg, N. Где reg - умножаемый регистр, N - количество разрядов сдвига. В отличие от MUL , SHL работает со всеми регистрами, а не только с A.
При сдвиге чисел вправо на N разрядов происходит обратный процесс - число целочисленно делится на 2 в степени N, где N - количество разрядов сдвига. Ассемблерная команда для сдвига вправо называется SHR (shift right) и имеет тот же формат, что SHL и так же работает с любым регистром, а не только с A (как DIV).
Используя эти знания, напишите собственную функцию умножения, пригодную для работы с любым регистром. Функция должна принимать первый и второй множитель и производить умножение на целую степень двойки с помощью SHL и выводить результат на output.
За основу разработки функции умножения можно взять следующий фрагмент, осуществляющий умножение числа в соответствии с таблицей степеней 2 до 6. В фрагменте число 3 умножится на 4.
Предусмотрите возможность умножения чисел для других степеней 2. Модифицируйте программу, так чтобы при переполнении регистра (>255) на output выводился корректный ответ.

```
MOV A, 3
MOV B, 4
CALL umn
HLT

umn:
 CMP B, 2
 JE sh1
 CMP B, 4
 JE sh2

sh1:
 SHL A, 1
 RET
sh2:
 SHL A, 2
 RET
```

Задача 4

Используя команды SHL и SHR, а также организацию цикла, модифицируйте разработанную Вами ранее функцию умножения, так чтобы умножение на степень двойки осуществлялось через определение позиции единицы в двоичной записи числа.

Для наглядности, рассмотрите двоичную запись степеней двойки до 6:

0000 0010 - 2
0000 0100 - 4
0000 1000 - 8
0001 0000 - 16
0010 0000 - 32
0100 0000 – 64

А что если сдвигать двоичную запись степени двойки вправо, пока не получится число, равное 1, инкрементировать в цикле счетчик степени от 1 до n, где n - количество итераций цикла со сдвигом, а затем, после цикла, сдвинуть первый множитель на n позиций влево.

Условием цикла может быть переход:

```
CMP reg, 1
JNE cycl
```

, где reg - регистр, в котором содержится сдвигаемое вправо число, cycl - метка входа в цикл

Задача 5

Используя разработанную ранее программу, напишите функцию для универсального умножения некоторого числа на некоторое другое число, могущее не являться целой степенью двойки.

Подход к реализации:

1) С помощью позиционирования единицы, найдите старшую степень двойки во втором множителе и сдвиньте первый множитель на нужное число разрядов.
2) Вычтите из исходного значения второго множителя степень двойки, которую вы нашли на 1 шаге
3) Повторите шаг 1) для уменьшенного второго множителя и найдите самую старшую степень двойки в нем, затем сдвиньте исходное значение первого множителя на нужное число разрядов влево.
4) Повторяйте шаги 2-3) до тех пор, пока вычитание на 2) шаге не даст нуль.
5) Просуммируйте, полученные на итерациях шага 3) произведения.
6) Выведите результат умножения на output

Пример.
Умножим число N на 20 (00010100b) (число 20 взято произвольно для примера, без ограничения общности).
20 не является целой степенью двойки.
На первом шаге мы получим, что старшей степенью двойки в этом числе является 4-ая степень или 16. Сдвинем N на 4 разряда влево: SHL reg, 4 , где reg - регистр, в который помещено число N
На втором шаге вычтем из 20 16, получим 4.
На третьем шаге найдем старшую степень двойки (2). Сдвинем N на 2 разряда влево: SHL reg1, 2 , где reg1 - регистр, в который помещено исходное значение N (не сдвинутое на предыдущей итерации!)
Вернемся к 2-му шагу. 4-4=0. Таким образом, итераций больше не будет. Переходим на 5 шаг
Суммируем частичные произведения, полученные нами: N*16 + N*4=N*(16+4)=N*20.
Выводим результат.

Для организации итераций необходимо запрограммировать цикл.

Рассмотрите возможности максимальной оптимизации использования регистров в этой программе. Вам нужно хранить исходное значение N и результаты частичного умножения на каждой итерации. Можно ли задействовать стек? Можно ли хранить промежуточные данные в оперативной памяти?