ПИН-22, команда 5: Надёжкина Анна, Трусов Михаил

Отчет

Лабораторная работа 7 (0111 = 7)

Флаги и условные команды. Ветвления и циклы

Операционная система ОС Windows, 64-разрядная ОС, соглашение о вызовах Microsoft x64 (x86-64).

Задание Л7.з1. *(Вариант 2)*

Вычислите сумму двух целых чисел $z = x + y$, используя команду add. Сформируйте w (таблица Л7.1), используя семейство команд $setCC$ и анализируя флаги состояния CF, OF, SF, ZF, AF, PF после вычисления z.

$$w = \begin{cases} 0, & \text{если не было знакового переполнения,} \\ 1, & \text{если было знаковое переполнение} \end{cases}$$

Программный код:

C++

```
#include <stdio.h>


extern "C" {
int task1(int a, int b);
}
int main() {
  int x = 0x7FFFFFFF;                    // 0111 1111 1111 1111 1111 1111 1111 1111
  int y = 0x0;                           // 0000 0000 0000 0000 0000 0000 0000 0000

  printf("%d\n", task1(x, y));

  x = 0x7FFFFFFF;                        // 0111 1111 1111 1111 1111 1111 1111 1111
  y = 0x1;                               // 0000 0000 0000 0000 0000 0000 0000 0001
  printf("%d\n", task1(x, y));
```

```
  x = 0x7FFFFFFE;                    // 0111 1111 1111 1111 1111 1111 1111 1110
  y = 0xl;                           // 0000 0000 0000 0000 0000 0000 0000 0001
  printf("%d\n", taskl(x, y));

  x = 0x7FFFFFFE;                    // 0111 1111 1111 1111 1111 1111 1111 1110
  y = 0x2;                           // 0000 0000 0000 0000 0000 0000 0000 0010
  printf("%d\n", taskl(x, y));
  return 0;
}
```

Ассемблер

```
.data
.text
.globl taskl
taskl:
   sub $56, %rsp

   add %ecx, %edx
   seto %al                    // seto: устанавливает биты, если флаг переполнения OF=1

   add $56, %rsp
   ret
```

Вывод:

Задание Л7.з2. (Вариант 1; Z = (x < 11))

Вычислите z для заданного целого беззнакового x; z принимает значение 1 либо 0, аналогично операторам сравнения C/C++

Программный код:

C++

```
#include <stdio.h>

extern "C" {
int task2(unsigned int x);
}

int main() {
  unsigned int x = 10;
  printf("%d\n", task2(x));

  x = 11;
  printf("%d\n", task2(x));

  x = -5;
  printf("%d\n", task2(x));

  x = 12;
  printf("%d\n", task2(x));
  return 0;
}
```

Ассемблер

```
.data
.text
.globl task2
task2:
   sub $56, %rsp

   cmp $11, %ecx                    // сравнение значение в регистре %ecx с 11
   jc good                          //выполняет переход к метке good, если CF установлен (%ecx<11)

   mov $0, %eax
   add $56, %rsp
   ret

good:
   mov $1, %eax
   add $56, %rsp
   ret
```

Вывод:

Задание Л7.з3.

Реализуйте Л7.з2 для целого знакового x

Программный код:

C++

```
#include <stdio.h>
extern "C" {
int task3(int x);
}
int main() {
    int x = 10;
    printf("%d\n", task3(x));
    x = 11;
    printf("%d\n", task3(x));
    x = -5;
    printf("%d\n", task3(x));
    x = 12;
    printf("%d\n", task3(x));
    return 0;
}
```

Ассемблер

```
.data
.text
.globl task3
task3:
    sub $56, %rsp
    cmp $11, %ecx
    jl good          // инструкция JL проверяет флаги SF и OF. Переход выполняется, если SF не равен OF.
```

```
    mov $0, %eax
    add $56, %rsp
    ret

good:
    mov $1, %eax
    add $56, %rsp
    ret
```

Вывод:

Задание Л7.з4. (Вариант 1; Двойной точности ($double$)

Реализуйте Л7.з2 для x с плавающей запятой (таблица Л7.3), используя AVX-команды сравнения $vcomisd/vcomiss$ (или их SSE-аналоги)

Программный код:

C++

```
#include <stdio.h>

extern "C" {
int task4(double x, double const);
}

int main() {
  double eleven = 11.0;
```

```
  double x = 10.56;
  printf("%d\n", task4(x, eleven));

  x = 11.00;
  printf("%d\n", task4(x, eleven));

  x = -5.54;
  printf("%d\n", task4(x, eleven));

  x = 12.32;
  printf("%d\n", task4(x, eleven));
  return 0;
}
```

Ассемблер

```
.data
.text
.globl task4
task4:
    sub $56, %rsp

    vcomisd %xmm1, %xmm0
    jc good

    mov $0, %rax
    add $56, %rsp
    ret
```

```
good:
    mov $1, %rax
    add $56, %rsp
    ret
```

Вывод: