«Национальный исследовательский университет ИТМО»

Основы профессиональной деятельности

Лабораторная работа 3

Вариант 3001

Бутвин Михаил, P3130

2023

Исходный код

Адрес	Значение	Мнемоника	Комментарий
253	0268	`array_st_pointer`	Адрес первого элемента массива
254	A000	`array_curr_pointer`	Адрес текущего элемента, обрабатываемого в цикле
255	E000	`loop_counter`	Счетчик цикла, равен длине массива
256	0200	`result`	Результат
257 +	0200	CLA	Начало программы. Очистка аккумулятора
258	EEFD	ST (IP-3)	Сохранить аккумулятор в `result`
259	AF04	LD #4	Загрузить 4 в аккумулятор
25A	EEFA	ST (IP-6)	Сохранить аккумулятор в `loop_counter`
25B	AEF7	LD (IP-9)	Загрузить значение `array_st_pointer` в аккумулятор
25C	EEF7	ST (IP-9)	Сохранить аккумулятор в `array_curr_pointer`
25D	AAF6	LD (IP-10)+	Начало тела цикла. `array_curr_pointer` содержит адрес текущего элемента массива, по этому адресу получаем элемент и загружаем в аккумулятор, инкрементируем указатель
25E	0480	ROR	Сдвигаем аккумулятор вправо, чтобы в флаг переноса перешел последний бит аккумулятора
25F	F401	BCS (IP+1)	Если перенос установлен (т.е. последний бит аккумулятор до сдвига был 1), переходим на 261 ячейку и продолжаем цикл, если нет, попадаем на джамп и уходим в конец цикла
260	CE04	JUMP (IP+4)	Переход в конец цикла
261	0400	ROL	Сдвигаем аккумулятор назад (зачем?все равно перезапишем следующей командой)
262	AEF3	LD (IP-13)	Загружаем в аккумулятор значение `result`
263	0700	INC	Инкрементируем аккумулятор
264	EEF1	ST (IP-15)	Сохраняем аккумулятор в `result`
265	8255	LOOP 0x255	Проверяем `loop_counter`. Если ноль, переходим в 0x267 завершаем программу, иначе попадаем на джамп и возвращаемся в начало цикла
266	CEF6	JUMP (IP-10)	Переход в начало цикла
267	0100	HLT	Прерывание

Адрес	Значение	Мнемоника	Комментарий
268	0280	`array`	4 идущих подряд значения массива
269	0200		
26A	225F		
26B	C257		

Описание программы

Программа начинает исполнение с адреса 0x257. В ячейках 0x268 - 0x26B хранится 4 элемента массива. `array_st_pointer` - это ссылка на начало. Она копируется в `array_curr_pointer` - указатель на текущий элемент. В `loop_counter` записываем длину массива, по этой переменной будет идти счетчик цикла.

В ячейках 0x25D - 0x266 находится цикл. Он проходит по каждому элементу цикла, увеличивая при каждом проходе `array_curr_pointer`. С помощью побитового сдвига и ветвления проверяем, есть ли единица в конце элемента и затем инкрементируем `result`. Получается, что программа считает количество нечетных элементов массива.

ОП и ОДЗ

Входные данные:

`array_st_pointer` (0x253), `array_curr_pointer` (0x254) - адрес первого элемента массива и текущего обрабатываемого элемента. Так как это адреса, то они должны быть в пределах от 0x000 до 0x7FF, т.к у нас 2048 ячеек памяти. Но в нашем случае положение массива фиксированно, так что `array_st_pointer` всегда 0x0268, а `array_curr_pointer` от 0x268 до 0x26B

`loop_counter` (0x255) - Счетчик цикла, задает, сколько значений массива пройдет цикл, значит должен быть <= (len(`array`) - `array_st_pointer`) и может быть положительным целым числом от 0 до 2026 (2048 ячеек всего, 22 под программу, остальное может занять массив)

> `array_st_pointer` и `loop_counter` не обязательно должны задавать массив целиком. Указав `array_st_pointer` не в начале и `loop_counter` не равный длине массива, мы можем обработать его срез

`array` (0x268 - 0x26B) - Четыре элемента массива, подряд хранящиеся в памяти. Могут быть интерпретированы как знаковые целые числа от -32768 до 32767

Выходные данные:

`result` (0x256) - Подсчитанное количество нечетных элементов. Может принимать значения от 0 до `loop_counter`

Ассемблерный код (просто так)

Basic computer assembly support for vs code

```
```

```
                        ORG                     0x253
array_st_pointer:       WORD                    $array
array_curr_pointer:     WORD                    ?
loop_counter:           WORD                    ?
result:                 WORD                    ?

START:                  CLA
                        ST                      result
                        LD                      #4
                        ST                      loop_counter
                        LD                      array_st_pointer
                        ST                      array_curr_pointer
loop_body:              LD                      (array_curr_pointer)+

                        BCC                     loop_end
                        ROL
                        LD                      result
                        INC
                        ST                      result
loop_end:               LOOP                    loop_counter
                        JUMP                    loop_body
                        HLT

                        ORG 0x268
array:                  WORD                  0x0280, 0x0200, 0x225F, 0xC257
```