ЗАДАНИЕ

Разработать программу для работы с элементами массива M, в которой:

1. Массив имеет следующие характеристики:

- адрес начала массива в памяти БЭВМ - 0x6c6;

- число измерений исходного массива - 1;

- количество элементов исходного массива - 23;

- каждый элемент является знаковым числом с разрядностью 21 бит;

- нумерация элементов начинается с 4;

- элементы хранятся в массиве по границам слов, нет необходимости в плотной упаковке;

2. Для элементов массива необходимо вычислить одно значение по правилам:

- агрегировать необходимо только для элементов массива с кратными 4-м i-индексами;

- из выбранных элементов необходимо вычислить максимальное значение и записать результат в память по адресу 0x400.

- Результатом является одно 32-х разрядное число!

Примечание: все числа представлены в десятичной системе счисления, если явно не указано иное.

РЕШЕНИЕ

ORG 0x6C6 ; тестовый массив

ARR: WORD 0x0000,0x10 ; <- #4 - загрузит 0xFFF00000

WORD 0xFFFF,0x1F ; <- #5 - проигнорирует, т.к. индекс 5 % 4 > 0

WORD 0xFFFF,0x1F

WORD 0xFFFF,0x1F

WORD 0xF000,0x10 ; <- #8 - загрузит 0xFFF0F000

WORD 0xFFFF,0x1F

WORD 0xFFFF,0x1F

WORD 0xFFFF,0x1F

WORD 0xF700,0x10 ; <- #12 - загрузит 0xFFF0F700 (корректно сравниваем младшие слова)

WORD 0xFFFF,0x1F

WORD 0xFFFF,0x1F

WORD 0xFFFF,0x1F

WORD 0x0000,0x00 ; <- #16 - загрузит 0x00000000

WORD 0xFFFF,0x1F

WORD 0xFFFF,0x1F

WORD 0xFFFF,0x1F

WORD 0x9060,0x0F ; <- #20 - загрузит 0x000F9060 <- [ответ]

WORD 0xFFFF,0x1F

WORD 0xFFFF,0x1F

WORD 0xFFFF,0x1F

WORD 0x9050,0x0F ; <- #24 - проигнорирует, т.к. меньше

WORD 0xFFFF,0x1F

WORD 0xFFFF,0x1F ; <- #26 - последний элемент массива (проигнорирует)

WORD 0xFFFF,0x1F

WORD 0xFFFF,0x0F ; <- #28 - проигнорирует, т.к. вышел за рамки массива

ORG 0x400

WORD 0x8000,0x0 ; здесь будет храниться результат (little-endian)

ORG 0x500

FUNC: LD &1 ; функция для поиска максимального элемента

INC

ST ARR\_A

LD RES

INC

ST RES\_A ; завершаем работу с индексами

CALL EXTEND

CMP (RES\_A)

BGE LOOK ; переход, если старшее слово результата не больше текущего элемента

RET

LOOK: BZS LOW

LD (ARR\_A) ; копируем текущий элемент в результат (т.к. старшее слово больше)

CALL EXTEND

ST (RES\_A)

LD -(RES\_A)

LD -(ARR\_A)

ST (RES\_A)

RET

LOW: LD -(RES\_A) ; сравниваем младшие слова (т.к. старшие слова равны)

LD -(ARR\_A)

CMP (RES\_A)

BCC QUIT

ST (RES\_A) ; копируем младшее слово текущего элемента в результат

QUIT: RET

ARR\_A: NOP ; адрес текущего элемента массива

RES\_A: NOP ; адрес результата

EXTEND: LD (ARR\_A) ; функция расширения знака старшего слова

AND SIGNCHK

BZC IFNEG

LD (ARR\_A)

AND ANDMASK ; маскируем мусор в старших битах

RET

IFNEG: LD (ARR\_A)

OR ORMASK ; расширяем '1' на старшие биты

RET

ORMASK: WORD 0xFFE0 ; маска (расширение отрицательного)

ANDMASK:WORD 0x1F ; маска (сохранение положительного)

SIGNCHK:WORD 0x10 ; маска знакового бита

START: CLA ; делаем программу перезапускаемой

ST VISIT

ST (RES)

LD (RES)+

LD INIT

ST (RES)

LD -(RES)

REP: LD VISIT ; начинаем обработку массива

DEC

CMP COUNT

BGE EXIT ; if (VISIT - 1 >= COUNT) { exit(); }

LD VISIT

ASL

ADD ARR\_INT

PUSH ; загружаем в стек адрес текущего элемента массива

CALL FUNC

POP

LD VISIT ; пропускаем элементы с номерами n % 4 > 0

ADD #4

ST VISIT

JUMP REP ; зацикливаемся

EXIT: HLT

VISIT: WORD 0 ; индекс посещённого элемента

COUNT: WORD 23 ; количество элементов в исходном массиве

RES: WORD 0x400 ; адрес младшего слова результата

INIT: WORD 0x8000 ; начальное значение старшего слова результата

ARR\_INT:LD $ARR ; адрес начала массива