

1. Решение задачи “Произведение *minimax*” на ассемблере

Дан вектор из 7 элементов. Найти произведение максимального и минимального элементов вектора.

Формат ввода:

a-вектор из 7 элементов.

Формат вывода: pr-произведение максимального и минимального элементов вектора.

Размер исходных данных и результата 1 байт.

Пример:

a dec : 2 3 4 5 6 7 8

pr dec : 16

ИИ: <https://www.bing.com/>

Ссылка на задачу: <https://dl.gsu.by/task.jsp?nid=2285605&cid=1335>

Диалог в скриншотах:

привет что ты знаешь об ассемблере версии i8086?

✓ Поиск: ассемблер версии i8086

✓ Создание ответов для вас...

Привет! Язык ассемблера представляет команды процессора в виде, доступном для чтения человеком. Он считается языком программирования низкого уровня ¹.

Ассемблер i8086 относится к архитектуре x86. Наиболее известными ассемблерами для операционной системы DOS являлись Borland Turbo Assembler (TASM), Microsoft Macro Assembler (MASM) и Watcom Assembler (WASM) ².

Ассемблер преобразует код на языке ассемблера в машинный код. Каждой мнемонике в языке ассемблера соответствует одна машинная инструкция ². Это делает ассемблирование более или менее однозначным и обратимым процессом ².

Пожалуйста, дайте мне знать, если вам нужна более конкретная информация или помощь с кодом на ассемблере i8086

Задайте любой вопрос...

0/4000

я сейчас покажу пример решения задачи и хочу чтобы ты потом помог мне решить похожую вот задача:

Дан вектор из 10 элементов. Найти сумму минимального и максимального элементов вектора.

JMP BEGIN

a DB 200, 50, 100, 80, 40, 99, 199, 131, 51, 111

summ DB 0

DB 0DH,0AH

BEGIN:

LEA SI, a

MOV CX, 10

MOV AL, [SI]

MOV BL, [SI]

FIND_MIN_MAX:

CMP AL, [SI]

JB UPDATE_MIN

CMP AL, [SI]

JB UPDATE_MIN

CMP BL, [SI]

JA UPDATE_MAX

INC SI

JMP NEXT_ITERATION

UPDATE_MIN:

MOV AL, [SI]

INC SI

JMP NEXT_ITERATION

UPDATE_MAX:

MOV BL, [SI]

INC SI

JMP NEXT_ITERATION

NEXT_ITERATION:

LOOP FIND_MIN_MAX



Введите любой вопрос...



0/4000 ➤

```
ADD AL, BL
MOV summ, AL
```

```
HALT:
JMP HALT ;$E
```

✓ Создание ответов для вас...

Спасибо за пример! Ваш код на ассемблере i8086 ищет минимальный и максимальный элементы в векторе из 10 элементов, а затем суммирует их.

Если вы хотите решить похожую задачу, мне нужно знать, что именно вы хотите изменить или добавить в этот код. Например, вы можете изменить размер вектора, тип операции (вместо суммы использовать умножение, например) или тип данных в векторе. Пожалуйста, уточните ваш запрос, и я буду рад помочь вам с этим.

Если вы хотите решить похожую задачу, мне нужно знать, что именно вы хотите изменить или добавить в этот код. Например, вы можете изменить размер вектора, тип операции (вместо суммы использовать умножение, например) или тип данных в векторе. Пожалуйста, уточните ваш запрос, и я буду рад помочь вам с этим.



2 из 30 ●

Дан вектор из 7 элементов. Найти произведение максимального и минимального элементов вектора.

✓ Создание ответов для вас...

Ваша задача - найти произведение минимального и максимального элементов в векторе из 7 элементов. Вот пример кода на ассемблере i8086, который выполняет эту задачу:

Assembly



```
JMP BEGIN
a DB 200, 50, 100, 80, 40, 99, 199 ; вектор из 7 элементов
min_max_product DW 0
DB 0DH,0AH
```

```
BEGIN:
LEA SI, a
MOV CX, 7
MOV AL, [SI]
MOV BL, [SI]

FIND_MIN_MAX:
CMP AL, [SI]
JB UPDATE_MIN
CMP BL, [SI]
JA UPDATE_MAX
INC SI
JMP NEXT_ITERATION
```

```
UPDATE_MIN:
MOV AL, [SI]
INC SI
JMP NEXT_ITERATION
```

```
UPDATE_MAX:
MOV BL, [SI]
INC SI
JMP NEXT_ITERATION
```

```
NEXT_ITERATION:
LOOP FIND_MIN_MAX

MOV AH, 0
MUL BL
MOV min_max_product, AX

HALT:
```



Задайте любой вопрос...



0/4000

HALT:

```
JMP HALT ;$E
```

Код, созданный ИИ. Внимательно просмотрите и используйте. [Дополнительные сведения о часто задаваемых вопросах](#).

В этом коде мы используем регистры `AL` и `BL` для хранения минимального и максимального элементов соответственно. После нахождения этих элементов, мы перемножаем их с помощью команды `MUL` и сохраняем результат в переменной `min_max_product`. Обратите внимание, что перед выполнением операции умножения мы обнуляем старший байт регистра `AX` с помощью команды `MOV AH, 0`, чтобы избежать переполнения.

Пожалуйста, дайте мне знать, если вам нужна дополнительная помощь!



3 из 30

все значения 1 байт

✓ Создание ответов для вас...



Задайте любой вопрос...



Понял, все значения в векторе занимают 1 байт. В этом случае, код будет выглядеть так:

Assembly



```
JMP BEGIN
a DB 200, 50, 100, 80, 40, 99, 199 ; вектор из 7 элементов
min_max_product DB 0
DB 0DH,0AH

BEGIN:
LEA SI, a
MOV CX, 7
MOV AL, [SI]
MOV BL, [SI]

FIND_MIN_MAX:
CMP AL, [SI]
JB UPDATE_MIN
CMP BL, [SI]
```



Задайте любой вопрос...



```
CMP BL, [SI]
JA UPDATE_MAX
INC SI
JMP NEXT_ITERATION

UPDATE_MIN:
MOV AL, [SI]
INC SI
JMP NEXT_ITERATION

UPDATE_MAX:
MOV BL, [SI]
INC SI
JMP NEXT_ITERATION

NEXT_ITERATION:
LOOP FIND_MIN_MAX

IMUL AX, BX
```



Задайте любой вопрос...



```
IMUL AX, BX
MOV min_max_product, AL
```

```
HALT:
JMP HALT ;$E
```

Код, созданный ИИ. Внимательно просмотрите и используйте. [Дополнительные сведения о часто задаваемых вопросах.](#)

В этом коде мы используем регистры `AL` и `BL` для хранения минимального и максимального элементов соответственно. После нахождения этих элементов, мы перемножаем их с помощью команды `IMUL` и сохраняем результат в переменной `min_max_product`.

Пожалуйста, дайте мне знать, если вам нужна дополнительная помощь!



4 из 30

imul работает с регистром al, он умножает al на другой регистр

✓ Создание ответов для вас...

Вы правы, моя ошибка. Команда `IMUL` в ассемблере i8086 используется для умножения регистра `AX` на другой регистр, а не `AL`. Вместо этого, мы можем использовать команду `MUL`, которая умножает `AL` на другой регистр и сохраняет результат в `AX`.

Вот исправленный код:

Assembly

```
JMP BEGIN
a DB 200, 50, 100, 80, 40, 99, 199 ; вектор из 7 элементов
min_max_product DB 0
DB 0DH,0AH

BEGIN:
LEA SI, a
MOV CX, 7
MOV AL, [SI]
MOV BL, [SI]
```



Задайте любой вопрос...

LEA SI, a

MOV CX, 7

MOV AL, [SI]

MOV BL, [SI]

FIND_MIN_MAX:

CMP AL, [SI]

JB UPDATE_MIN

CMP BL, [SI]

JA UPDATE_MAX

INC SI

JMP NEXT_ITERATION

UPDATE_MIN:

MOV AL, [SI]

INC SI

JMP NEXT_ITERATION



Задайте любой вопрос...



```
UPDATE_MAX:
MOV BL, [SI]
INC SI
JMP NEXT_ITERATION

NEXT_ITERATION:
LOOP FIND_MIN_MAX

MOV AH, 0
MUL BL
MOV min_max_product, AL

HALT:
JMP HALT ;$E
```

Код, созданный ИИ. Внимательно просмотрите и используйте. [Дополнительные сведения о часто задаваемых вопросах.](#)

В этом коде мы используем команду `MUL` для умножения регистров `AL` и `BL`, и результат сохраняется в `AL`. Затем мы сохраняем результат в переменной `min_max_product`.



Задайте любой вопрос...



```
MUL BL
MOV min_max_product, AL
```

```
HALT:
JMP HALT ;$E
```

Код, созданный ИИ. Внимательно просмотрите и используйте. [Дополнительные сведения о часто задаваемых вопросах.](#)

В этом коде мы используем команду `MUL` для умножения регистров `AL` и `BL`, и результат сохраняется в `AL`. Затем мы сохраняем результат в переменной `min_max_product`.

Спасибо за ваше замечание, и извините за ошибку в предыдущем коде. Пожалуйста, дайте мне знать, если вам нужна дополнительная помощь!



5 из 30 ●



Как я могу изменить размер вектора?

Можете ли вы объяснить, как работает этот код?



Задайте любой вопрос...



0/4000



