**Программирование микропроцессора Intel 8086.**

Инструкции по передаче. Арифметические инструкции. Логические инструкции, перемещения и вращения

**Цель работы**

а) Элементы ассемблера.

б) Арифметические инструкции и инструкции передачи

в) Логические инструкции, перемещения и вращения.

**1. Элементы языка ассемблера для микропроцессора Intel 8086**

***1.1. Типы данных, используемые микропроцессором Intel 8086***

Внутреннее представление информации (инструкции, данные, адреса, команды и т. Д.) Выполнено в двоичном формате. Таким образом, основные типы данных (группы битов), используемые микропроцессором Intel 8086, имеют следующие размеры и названия:

- бит = двоичный символ,

- байт = группа из 8 бит,

- слово (Word) = 16-битная группа = 2 байта (MSB = старший байт, LSB = младший байт),

- Двойное слово = Группа двойных слов = 32 байта = 4 байта = 2 слова (MSW = верхнее слово, LSW = нижнее слово).

Двойное слово - это тип даты, необходимый для хранения логического адреса, поэтому его также называют «указателем».

Типом производных данных является запись битов, состоящая из 8 или 16 битов (байтов или слов), состоящих из битовых полей разной длины с разным значением. Среди типов составных данных, необходимых для сложных структур данных, наиболее часто используется строка данных, которая позволяет хранить массивы (векторы, матрицы и т. Д.). Другой составной тип - это структура, которая представляет собой множество разнородных данных (байтов, слов и т. Д.).

***1.2. Структура программы на ассемблере для микропроцессора Intel 8086***

Ниже приводится пример программы на языке ассемблера для микропроцессора Intel 8086.

Программа выполняет сумму N = 5:

DSEG SEGMENT ; сегмент данных

TAB LABEL BYTE ; ярлык TAB

TABW DW 5 DUP(7000H) ; строка слов (массив) TABW

REZ DB ? , ? , ? , ? ; байтовая строка (массив) REZ

ZERO EQU 0 ; постоянная ZERO

DSEG ENDS

STIVA SEGMENT ; сегмент стека

DW 40 DUP(?)

VIRF LABEL WORD ; метка VIRF

STIVA ENDS

CSEG SEGMENT ; сегмент кода (программа)

START: ; начальная метка программы

MOV AX, DSEG ; инициализации

MOV DS, AX

MOV ES, AX

MOV AX, STIVA

MOV SS, AX

MOV SP, OFFSET VIRF ; инициализация указателя стека

MOV AX, 0 ; инициализация использ. регистров MOV DX, 0

MOV CX, LENGTH TABW ; CX = 5 = длина TABW

MOV BP, SIZE TABW ; BP = 10 = 5\*2 = размерность TABW

NEXT:

SUB BP, TYPE TABW ; BP = BP - 2

ADD AX, DS: TABW [BP] ; AX = AX + текущее слово из TABW

ADC DX, ZERO ; DX = DX + предыдущий транспорт

LOOP NEXT ; CX = CX-1

; если CX <> 0 переходит к NEXT

; в противном случае переходите к след. инструкциям

MOV WORD PTR REZ, AX ; хранение LSW под адресом REZ

MOV WORD PTR REZ + 2 , DX ; хранение MSW под адресом REZ+2

MOV AH, 4CH

; функция DOS 4CH (завершение процесса)

INT 21H ; прерывание SW – вызов функции DOS 4CH

CSEG ENDS

END START ; конец программы

***1.3. Инструкции на языке ассемблера для микропроцессора I8086***

Программа на ассемблере состоит из нескольких строк исходного кода. Исходная строка

состоит из следующих полей:

| Метка: | Код операции Операнды мгновенного типа ; Комментарий |

 (мнемоника)

где между | | дополнительные элементы включены. Поле метки является необязательным и представляет собой символическое имя адреса в памяти, где находится числовой код инструкции. Мнемоника кода операции - это символическое имя инструкции. Поле операнда может содержать два операнда, один или ни одного, в зависимости от типа инструкции. Операнды могут быть данными, на которые действуют инструкции, адреса или другие вспомогательные инструкции. Комментарий не обязателен. Он служит только для повышения разборчивости программы. При сборке текст комментария игнорируется ассемблером.

***1.4. Операторы ассемблера***

Операции также могут выполняться с операндами инструкций программой ассемблера. Эти типы операций задаются инструкциями и выполняются ассемблером после сборки исходного текста. Инструкции, используемые ассемблером TASM, бывают нескольких типов:

**Операторы изменения значений:**

а) Арифметические инструкции (для элементарных арифметических операций: +, -, \*, /, MOD).

Пример:

MOV WORD PTR REZ + 2, DX

Ассемблер выполняет операцию сборки между адресом REZ (компонент типа смещения) и 2, результат используется объектным кодом.

б) Инструкции индексации [ ], позволяющие использовать для адресации.

Пример:

ADD AX, DS: TABW [BP]

Ассемблер использует регистр BP для вычисления фактического адреса TABW + BP, необходимого для получения физического адреса из логического адреса сегмента (DS): смещение (TABW + BP).

***1.5. Директивы ассемблера***

Программа, написанная на языке ассемблера, вместе с фактическими инструкциями и псевдо- инструкциями (директивами), адресована программе ассемблера путем изменения способа ее работы или разрешения спецификации данных.

1. Директивы, определяющие используемые сегменты:

а) Директива SEGMENT - используется для обозначения начала сегмента (например, DSEG, CSEG, STIVA);

Пример:

DSEG SEGMENT

определяет сегмент данных под названием DSEG,

STIVA SEGMENT

определяет сегмент стека (здесь не используется), называемый STIVA,

CSEG SEGMENT

определяет сегмент кода (программу) под названием CSEG.

б) Директива ENDS - используется для обозначения конца сегмента;

Пример:

DSEG ENDS

объявляет конец сегмента данных DSEG,

STIVA ENDS

объявляет конец сегмента стека STIVA,

CSEG ENDS

объявляет конец сегмента кода CSEG.

2. Директива EQU разрешает объявление констант. Объявленные таким образом константы заменяются фактическими числами во время сборки. Для них не выделяется места в памяти.

Пример:

ZERO EQU 0

Ассемблер заменяет в программе константу ZERO числовым значением 0.

3. Следующие директивы используются для объявления переменных:

a) Директива DB (Define Byte) объявляет байты или строки байтов.

Пример:

REZ DB? ,? ,? ,?

объявляет переменную REZ типа байтовой строки, состоящую из 4 неинициализированных байтов

б) Директива DW (Define Word) объявляет слова или цепочки слов.

Пример:

TABW DW 5 DUP (7000H)

объявляет строковую переменную TABW, состоящую из 5 идентичных слов, инициализированных с помощью 7000H DW 40 DUP (?)

объявляет строку из 40 неинициализированных безымянных слов (простое присвоение)

в) Директива DD (Define Double-word) объявляет двойные слова или строки из двойных слов. Переменные определяются как резиденты по определенному относительному адресу (смещению) в пределах определенного сегмента и характеризуются типом данных. Замечено, что для резервирования памяти неинициализированных переменных используется инструкции:

? = резервирование области памяти для неинициализированной переменной и для указания нескольких значений переменных используется инструкции:

DUP = указанное количество повторов (DUPlicate).

4. Объявление меток, используемых для ссылки на данные из сегментов данных или стека (метки адресов переменных), выполняется с помощью директивы LABEL, возможно, с указанием атрибута.

Пример:

TAB LABEL BYTE

определяет метку с именем TAB перед переменной TABW, разрешая доступ к байту переменной TABW (вариант для BYTE PTR!),

VIRF LABEL WORD

определяет метку VIRF.

Ярлыки (как и переменные) представляют собой символьные имена адресов. Для них характерно определенное смещение внутри сегмента. Метки обычно обозначают инструкции. В этом случае на метки можно ссылаться в других инструкциях по выполнению переходов в программе. Если ссылки на метку делаются внутри сегмента, в котором она определена («домашний сегмент»), то говорят, что она имеет атрибут NEAR. На метку можно ссылаться в инструкции другого логического сегмента, если она несет атрибут FAR. Атрибуты метки установлены на их определение. Для объявления меток в сегменте программы (меток адресов инструкции) также используется инструкции:

Пример:

START:

объявляет начальную метку исполняемого кода

NEXT:

объявляет тег NEXT, используемый для итераций

5. Директива END, используемая для объявления конца программы (кода).

Пример:

END START

указывает ассемблеру, что программа, запущенная с метки START, заканчивается.

6. Директива RECORD, используемая для определения записей данных. Формат инструкции, определяющей запись:

имя\_записи RECORD имя\_поля: выражение | = выражение '| |, .. |

где:

имена полей (field\_name, ...) уникальны;

выражение expression оценивается как целочисленная константа от 1 до 16 и определяет количество бит (длину) поля; сумма длин должна быть меньше или равна 16, соответственно 8; выражение 'является необязательным и служит начальным значением поля; |, ... | указывает на необязательное повторение field\_name: expression | = выражение '|

Пример 1:

CHIPS RECORD RAM: 7, EPROM: 4, ROM: 5

определяет 16-битную запись, состоящую из 3 полей с именами RAM, EPROM и ROM длиной 7, 4 и 5 бит соответственно.

Пример 2:

CHIPS RECORD RAM: 7 = 4, EPROM: 4 = 2, ROM: 5 = 0

определяет 16-битную запись, состоящую из 3 полей с именами RAM, EPROM и ROM длиной 7, 4 и 5 бит соответственно, определяя начальные значения полей.

7. Директива STRUCT, используемая для определения структур. Формат заявления, определяющего структуру, следующий:

structure\_name STRUCT

| field\_name | {DB | DW | DD} выражение | |, .. |

structure\_name ENDS

где:

имена полей (field\_name, ...) уникальны;

|, ... | указывает на необязательное повторение field\_name {DB | DW | DD} выражение

Пример:

PROCES STRUCT

STARE DB 0

VAL\_CRT DW ?

PROCES ENDS

**2. Операнды инструкций микропроцессора Intel 8086**

Микропроцессор Intel 8086 использует несколько способов адресации данных, обычно называемых операндами (источник или место назначения), в зависимости от того, где они размещены. Операнды могут содержаться в регистрах, в памяти, в коде инструкции или в портах ввода / вывода. Операнды в регистрах благодаря своему внутреннему размещению позволяют выполнять инструкции, которые их используют, быстрее и больше не являются необходимыми передачами с памятью или портами. Регистры (8 или 16 битов) могут быть операндами источника, операндами назначения или операндами источника и назначения. Операнды непосредственного типа - это 8- или 16-битные постоянные данные, содержащиеся в коде инструкции после кода операции. Доступ к этим операндам довольно быстрый, поскольку они загружаются в очередь инструкции блоком интерфейса вместе с инструкцией. Ограничения операндов мгновенного типа связаны с тем, что они могут быть только постоянными значениями и могут служить только исходными операндами. Доступ к операндам памяти осуществляется медленнее, поскольку они необходимы: сначала вычисление фактического адреса операнда (EA = смещение), затем вычисление физического адреса и, наконец, их передача.

Общая формула расчета советника (смещение):

EA = (BX / BP) \* + (SI / DI) \* + (D8 / D16) \*

где: \* обозначает необязательный термин, а

/ разделяет варианты термина.

Примечание: адресное пространство портов 0 ... FFFFH (64ko)

***2.1. Мгновенные операнды (мгновенные адресация)***

Операнд явно указан в коде инструкции после кода операции, находящегося в очереди инструкций.

Пример 1. В инструкции:

ADD AL, 6

второй операнд - числовая константа 6 мгновенного типа.

Пример 2. В инструкции:

ADC ALFA, AX

первый операнд - символьная константа ALFA непосредственного типа.

***2.2. Операнды регистров (прямое обращение к регистру)***

Операнд находится в регистре, указанном в коде инструкции, в коде операции.

Пример. В инструкции:

ADD AL, BL

оба операнда относятся к регистровому типу.

***2.3. Операнды памяти (адресация памяти)***

1. Прямая адресация к памяти.

Операнд находится в основной памяти (MP) со смещением, указанным в коде инструкции (после кода операции). Фактический адрес указан в коде инструкции.

Пример. В инструкции:

ADD AL, VAR

второй операнд - это переменная VAR (замененная при сборке смещением переменной VAR), объявленная директивой. Операнд - это содержимое переменной VAR в памяти.

2. Косвенная адресация к памяти через регистр.

Операнд находится в основной памяти (MP) по смещению, расположенному в базовом регистре (BX, BP) или в индексе (SI, DI), указанном в коде инструкции, в коде операции.

Пример 1. В инструкции:

ADD AX, [BX]

второй операнд - это пара байтов (слово) в основной памяти в текущем сегменте данных (заданном DS), расположенных по смещению, содержащемуся в регистре BX.

Пример 2. В инструкции:

ADD [SI], AL

первый операнд - это байт в MP в текущем сегменте данных (заданном DS), расположенный по смещению, содержащемуся в регистре SI.

3. Косвенная адресация к памяти путем перемещения регистра.

Операнд находится в основной памяти MP со смещением, вычисленным как сумма содержимого базового регистра или индекса, указанного в коде операции, и следующего перемещения в коде инструкции после кода операции.

Пример 1. В инструкции:

ADD AX, DS: [BP + 2]

второй операнд - это слово в MP из текущего сегмента данных (явно заданного DS) со смещением, равным сумме между содержимым регистра BX и сдвигом 2 (мгновенного типа).

Пример 2. В инструкции:

ADD TAB[DI], AL

первый операнд - это слово в MP в текущем сегменте данных (заданном DS) со смещением, равным сумме между содержимым регистра DI и сдвигом TAB (мгновенный тип).

4. Косвенная адресация памяти через два регистра.

Операнд находится в основной памяти MP со смещением, вычисленным как сумма содержимого базового и индексного регистров, указанных в коде операции.

Пример. В инструкции:

ADD AX, [BX] [SI]

второй операнд - это слово из MP из текущего сегмента данных со смещением, равным сумме содержимого регистров BX и SI.

5. Косвенная адресация к памяти двумя движущимися регистрами.

Операнд находится в основной памяти MP со смещением, вычисленным как сумма содержимого базового и индексного регистров, указанных в коде операции, и смещения, следующего за кодом операции.

Пример. В инструкции:

MOV MATR [BX] [SI], AX

первый операнд - это слово из MP, из текущего сегмента данных, расположенного со смещением, равным сумме содержимого регистров BX и SI со смещением MATR (тип смещения).

***2.4. Операнды с портов ввода-вывода (адресация портов)***

1. Прямая адресация портов (для 8-битных адресов в диапазоне 0..255).

Операнд находится в порту по адресу, указанному в инструкции в соответствии с кодом операции.

Пример 1. В инструкции:

IN AX, 20H

второй операнд - это входной порт, расположенный по числовому адресу 20H (мгновенный тип).

Пример 2. В инструкции:

OUT PORT2, AL

первый операнд - это выходной порт, расположенный по символьному адресу PORT2 (мгновенный тип).

2. Косвенная адресация портов через регистр DX.

Операнд находится в порту по адресу, указанному в регистре DX.

Пример. В инструкции:

OUT DX, AL

первый операнд - это выходной порт, расположенный в порту по адресу, содержащемуся в DX.

**3. Инструкции передачи**

Команды передачи между регистрами или между регистром и памятью выполняют операции присваивания (копирования) и перестановки. Операции передачи также могут выполняться между регистрами и портами.

***3.1. Инструкция присвоения (копирования) имеет вид:***

MOV операнд1, операнд2

и эффект:

операнд1 = операнд2

Замечено, что значение, в котором выполняется присвоение, - справа налево (это означает, что обычно используется для записи математических функций и выражений). Таким образом, можно сказать, что форма инструкции такая:

Пункт назначения MOV, источник

Пример:

MOV AX, BX

копирует в регистр AX значение, содержащееся в регистре BX

***3.2 Инструкции перестановки имеет вид:***

XCHG операнд1, операнд2

и эффект:

temp = операнд 1

операнд1 = операнд2

операнд 2 = temp

Таким образом, два операнда изменяют свое содержимое между собой, причем операнды являются как источником, так и получателем. Пример:

MOV CX, BX

копирует в регистр CX значение, содержащееся в регистре BX, а в регистр BX - начальное значение CX

***3.3*** ***Инструкции чтения из порта имеет вид:***

IN аккумулятор, порт

и эффект:

 аккумулятор = порт

***3.4 Инструкции записи в порт имеет вид:***

 OUT порт, аккумулятор

и эффект:

порт = аккумулятор

**4. Арифметические инструкции**

Арифметические инструкции микропроцессора Intel 8086 используют 1 или 2 операнда.

***4.1. Арифметические инструкции, использующие 2 операнда***

Инструкции, которые используют 2 операнда: сложение (с битом Carry из предыдущей операции или без него), вычитание (с битом Carry из предыдущей операции или без него), умножение, деление. В следующем примере операнд1 - это источник и место назначения, а операнд2 - только источник.

1. Инструкция сложения имеет вид:

ADD операнд1, операнд2

и эффект: операнд1 = операнд1 + операнд2

Пример (изначально AX = 7FFFH и BX = 8000H):

ADD BX, DX

Эффект:

1000 0000 0000 0000 B = 8000 H +

1100 0000 0000 0000 B = C000 H

--------------------------------------------------

 (CF = 1) 0100 0000 0000 0000 B = 4000 H

Значения после выполнения инструкции: BX = 4000H, CF = 1 (транспорт).

2. Инструкция сложения с остатком из предыдущей операции имеет вид:

ADC операнд1, операнд2

и эффект: операнд1 = операнд1 + операнд2 + CF.

Пример (изначально AX = 7FFFH, CX = 4000H и CF = 1):

ADD AX, CX

Эффект:

0111 1111 1111 1111 B = 7FFF H +

 0100 0000 0000 0000 B = 4000 H +

 1 B = 1 H

--------------------------------------------------

 (CF = 0) 1100 0000 0000 0000 B = C000 H

Значения после выполнения инструкции: AX = C000H, CF = 0.

3. Инструкция вычитания имеет вид:

SUB операнд1, операнд2

и эффект: операнд1 = операнд1 - операнд2

Пример (изначально BX = 8000H и DX = 0C000H):

SUB BX, DX

Эффект:

1000 0000 0000 0000 B = 8000 H -

1100 0000 0000 0000 B = C000 H

--------------------------------------------------

 (CF = 1) 1100 0000 0000 0000 B = C000 H

Значения после выполнения инструкции: BX = C000H, CF = 1

4. Инструкция вычитания с остатком из предыдущей операции имеет вид:

SBB операнд1, операнд2

и эффект: операнд1 = операнд1 - операнд2 - CF

Пример (изначально AX = 7FFFH, CX = 4000H и CF = 1):

SBB AX, CX

0111 1111 1111 1111 B = 7FFF H -

 0100 0000 0000 0000 B = 4000 H -

 1 B = 1 H

--------------------------------------------------

 (CF = 0) 0011 1111 1111 1110 B = 3FFE H

Значения после выполнения инструкции: AX = 3FFEH, CF = 0.

5. Инструкция умножения беззнаковых чисел имеет вид:

MUL operand

и эффект:

- для 8-битных операндов: AX = AL \* операнд

- для 16-битных операндов: DX, AX = AX \* операнд

6. Инструкции умножения между знаковыми числами имеет вид:

IMUL operand

и эффект:

- для 8-битных операндов: AX = AL \* операнд

- для 16-битных операндов: DX, AX = AX \* операнд

7. Инструкция деления беззнаковых чисел имеет вид:

DIV operand

и эффект:

- для 8-битных операндов: AL = AX / операнд

 AH = операнд AX MOD (остаток)

- для 16-битных операндов: AX = DX, AX / операнд

 DX = DX, операнд AX MOD

8. Инструкция деления на числа со знаком имеет вид:

IDIV операнд

и эффект:

- для 8-битных операндов: AL = AX / операнд

AH = операнд AX MOD

- для 16-битных операндов: AX = DX, AX / операнд

 DX = DX, операнд AX MOD

***4.2. Арифметические инструкции с использованием 1 операнда***

Инструкции, которые используют 1 операнд (источник и место назначения): инкремент, декремент, отрицание (арифметика в дополнении до 2).

1. Инструкции приращения имеет вид:

INC операнд

и эффект: операнд = операнд + 1

Пример (начальный DI = 0FFFFH):

INC DI

Эффект:

1111 1111 1111 1111 B = FFFF H +

0000 0000 0000 0001 B = 1 H

--------------------------------------------------

 (CF = 1) 0000 0000 0000 0000 B = 0000 H

Значение после выполнения инструкции: DI = 0000H, CF = 1 (транспорт).

2. Инструкции уменьшения имеет вид:

DEC операнд

и эффект: операнд = операнд - 1

Пример (изначально SI = 0001H):

DEC SI

Эффект:

0000 0000 0000 0001 B = 0001 H -

0000 0000 0000 0001 B = 1 H

--------------------------------------------------

 (CF = 0) 0000 0000 0000 0000 B = 0000 H

Значение после выполнения инструкции: SI = 0000H, CF = 0.

3. Инструкции отрицания имеет форму:

NEG операнд

и эффект: операнд = (0) - операнд

Пример 1 (изначально SI = 0001H):

NEG SI

Эффект:

(0000 0000 0000 0000 B = 0000 H)

 - 0000 0000 0000 0001 B = 0001 H

--------------------------------------------------

 (CF = 1) 1111 1111 1111 1111 B = FFFF H

Значение после выполнения инструкции: SI = FFFFH, CF = 1 (транспорт).

Пример 2 (изначально DI = 0FFFFH):

NEG DI

Эффект:

(0000 0000 0000 0000 B = 0000 H)

 - 1111 1111 1111 1111 B = FFFF H

--------------------------------------------------

 (CF = 1) 0000 0000 0000 0001 B = 0001 H

Значение после выполнения инструкции: DI = 0001H, CF = 1 (транспорт).

Пример 3 (начальный AX = 7FFFH):

NEG AX

Эффект:

(0000 0000 0000 0000 B = 0000 H)

 - 0111 1111 1111 1111 B = 7FFF H

--------------------------------------------------

 (CF = 1) 1000 0000 0000 0001 B = 8001 H

Значение после выполнения инструкции: AX = 8001H, CF = 1 (транспорт).

Пример 4 (начальный BX = 8000H):

NEG BX

Эффект:

(0000 0000 0000 0000 B = 0000 H)

 - 1000 0000 0000 0000 B = 8000 H

--------------------------------------------------

 (CF = 1) 1000 0000 0000 0000 B = 8000 H

Значение после выполнения инструкции: BX = 8000H, CF = 1 (транспорт).

Пример 5 (изначально CX = 4000H):

NEG CX

Эффект:

(0000 0000 0000 0000 B = 0000 H)

 - 0100 0000 0000 0000 B = 4000 H

--------------------------------------------------

 (CF = 1) 1100 0000 0000 0000 B = C000 H

Значение после выполнения инструкции: CX = C000H, CF = 1 (транспорт).

Пример 6 (исходный DX = 0C000H):

NEG DX

Эффект:

(0000 0000 0000 0000 B = 0000 H)

 - 1100 0000 0000 0000 B = C000 H

--------------------------------------------------

 (CF = 1) 0100 0000 0000 0000 B = 4000 H

Значение после выполнения инструкции: DX = 4000H, CF = 1 (транспорт).

**5. Логические инструкции**

***5.1. Логические инструкции, использующие 2 операнда***

инструкции, которые используют 2 операнда: функция и логика между битами операнда (AND), функция или логика между битами операнда (OR), функция или исключительно логическая между битами операнда (XOR).

В следующем примере операнд1 - это источник и место назначения, а операнд2 - только источник.

1. Инструкции AND имеет вид:

AND операнд1, операнд2

и эффект: операнд1 = операнд1 AND операнд2

Пример (изначально AX = 5555H и BX = 6666H):

AND AX, BX

Эффект: 0101 0101 0101 0101 B = 5555 H x

0110 0110 0110 0110 B = 6666 H

-------------------------------------------------

0100 0100 0100 0100 B = 4444 H

Значение после выполнения инструкции: AX = 4444H.

2. Инструкции OR имеет вид:

OR операнд1, операнд2

и эффект: операнд1 = операнд1 OR операнд2

Пример (изначально AX = 5555H и BX = 6666H):

OR AX, BX

Эффект:

0101 0101 0101 0101 B = 5555 H +

0110 0110 0110 0110 B = 6666 H

------------------------------------------------

0111 0111 0111 0111 B = 7777 H

Значение после выполнения инструкции: AX = 7777H.

3. Инструкции XOR имеет вид:

XOR операнд1, операнд2

и эффект: operand1 = operand1 XOR operand2

Пример (изначально AX = 5555H и BX = 6666H):

XOR AX, BX

Эффект:

0101 0101 0101 0101 B = 5555 H +

0110 0110 0110 0110 B = 6666 H

------------------------------------------------

0011 0011 0011 0011 B = 3333 H

Значения, следующие за выполнением инструкции AX = 3333H.

***5.2. Логические инструкции, использующие 1 операнд***

Инструкции, который использует 1 операнд (источник и место назначения), является логическим отрицанием (NOT)

Биты операнда (представление в дополнении к 1 - C1).

Далее операнд - это источник и место назначения.

Инструкции NOT имеет вид:

NOT операнд

и эффект: операнд = NOT операнд

Пример 1 (изначально SI = 0001H):

NOT SI

Эффект:

0000 0000 0000 0001 B = 0001 H \

--------------------------------------------------

1111 1111 1111 1110 B = FFFE H

Значение после выполнения инструкции: SI = FFFEH.

Пример 2 (изначально DI = 0FFFFH):

NOT DI

Эффект:

1111 1111 1111 1111 B = FFFF H \

--------------------------------------------------

0000 0000 0000 0000 B = 0000 H

Значение после выполнения инструкции: DI = 0000H.

Пример 3 (начальный AX = 7FFFH):

NOT AX

Эффект:

0111 1111 1111 1111 B = 7FFF H \

--------------------------------------------------

1000 0000 0000 0000 B = 8000 H

Значение после выполнения инструкции: AX = 8000H.

Пример 4 (начальный BX = 8000H):

NOT BX

Эффект:

1000 0000 0000 0000 B = 8000 H \

--------------------------------------------------

0111 0111 0111 0111 B = 7FFF H

Значение после выполнения инструкции: BX = 7FFFH.

Пример 5 (изначально CX = 4000H):

NOT CX

Эффект:

0100 0000 0000 0000 B = 4000 H \

--------------------------------------------------

1100 0000 0000 0000 B = C000 H

Значение после выполнения инструкции: CX = C000H.

Пример 6 (исходный DX = 0C000H):

NOT DX

Эффект:

1100 0000 0000 0000 B = C000 H \

--------------------------------------------------

0011 1111 1111 1111 B = 3FFF H

Значение после выполнения инструкции: DX = 3FFFH.

**6. Инструкции перемещения и вращения**

***6.1. Инструкции перемещения***

Команды перемещения выполняют операции с одним операндом (источником и получателем).

Далее операнд - это источник и место назначения.

1. Команда логического сдвига влево имеет вид:

SHL-операнд, 1

и эффект: операнд = операнд, сдвинутый влево на одну позицию

или операнд SHL, CL

соответственно операнд = операнд, сдвинутый влево с (CL) поз.

Пример 1 (изначально AX = YYYY):

SHL CL, 1

Эффект: 1

1010 1010 1010 1010 B = AAAA H (<-----)

------------------------------------------------------------

0101 0101 0101 0100 B = 5 5 5 4 H

Значение после выполнения инструкции: AX = 5554H, CF = 1.

Пример 2 (изначально AX = YYYY и CL = 4):

SHL AX, CL

Эффект: (CL) = 4

1010 1010 1010 1010 B = AAAA H (<-----)

------------------------------------------------------------

1010 1010 1010 0000 B = AAA0 H

Значение после выполнения инструкции: AX = AAA0H, CF = 0.

Замечено, что логический сдвиг влево на n позиций имеет тот же эффект, что и умножение на 2n.

Таким образом, для первого примера имеем (n = 1, 2n = 21 = 2):

AAAA H \* 2 H = 1 5 5 5 4 H

а для второго (n = 4, 2n = 24 = 16 = 10 H):

AAAA H \* 10 H = AAAA0 H

Также замечено, что в первом случае правильный результат содержится в CF (1) и AX

(5554H), а во втором случае теряются первые биты результата.

2. Команда логического сдвига вправо имеет вид:

Операнд SHR, 1

и эффект: операнд = операнд, сдвинутый вправо на одну позицию

или операнд SHR, CL

соответственно операнд = операнд, сдвинутый вправо с (CL) поз.

Пример 1 (изначально AX = YYYY):

SHR AX, 1

Эффект: 1

1010 1010 1010 1010 B = AAAA H (----->)

------------------------------------------------------------

0101 0101 0101 0101 B = 5 5 5 5 H

Значение после выполнения инструкции: AX = 5555H, CF = 0.

Пример 2 (изначально AX = YYYY и CL = 4):

SHR AX, CL

Эффект: (CL) = 4

1010 1010 1010 1010 B = AAAA H (----->)

------------------------------------------------------------

0000 1010 1010 1010 B = 0AAA H

Значение после выполнения инструкции: AX = 0AAAH, CF = 1.

Замечено, что логический сдвиг вправо на n позиций имеет тот же эффект, что и деление на 2n.

Таким образом, для первого примера имеем (n = 1, 2n = 21 = 2):

AAAA H : 2 H = 5 5 5 5 H

а для второго (n = 4, 2n = 24 = 16 = 10 H):

AAAA H : 10 H = AAA H

3. Инструкции по арифметическому перемещению влево имеет вид:

Операнд SAL, 1

и эффект: операнд = операнд, сдвинутый влево на одну позицию

или операнд SAL, CL

соответственно операнд = операнд, сдвинутый влево с (CL) поз.

Пример 1 (изначально AX = YYYY):

SAL AX, 1

Эффект: 1

1010 1010 1010 1010 B = AAAA H (<-----)

-----------------------------------------------------------

0101 0101 0101 0100 B = 5 5 5 4 H

Значение после выполнения инструкции: AX = 5554H, CF = 1.

Пример 2 (изначально AX = YYYY и CL = 4):

SAL AX, CL

Эффект: (CL) = 4

1010 1010 1010 1010 B = AAAA H (<-----)

------------------------------------------------------------

1010 1010 1010 0000 B = AAA 0 H

Значение после выполнения инструкции: AX = AAA0H, CF = 0.

Замечено, что действие этого инструкции идентично действию инструкции логического сдвига влево !.

4. Инструкции арифметического перемещению вправо (с сохранением знака) имеет вид:

Операнд SAR, 1

и эффект: операнд = операнд арифметическое перемещение в право с 1 поз.

или операнд SAR, CL

соответственно операнд = операнд арифметическое перемещение в право с (CL) pos

Пример 1 (изначально AX = YYYY):

SAR AX, 1

Эффект: 1

1010 1010 1010 1010 B = AAAA H (----->)

------------------------------------------------------------

1101 0101 0101 0101 B = D5 5 5 H

Значение после выполнения инструкции: AX = D555H, CF = 0.

Пример 2 (изначально AX = YYYY и CL = 4):

SAR AX, CL

Эффект: (CL) = 4

1010 1010 1010 1010 B = AAAA H (----->)

------------------------------------------------------------

1111 1010 1010 1010 B = FAAA H

Значение после выполнения инструкции: AX = FAAAH, CF = 1.

***6.2. Инструкции вращения***

Команды вращению выполняют операции с одним операндом (источником и получателем).

Далее операнд - это источник и место назначения.

1. Инструкции вращения влево без CF имеют вид:

Операнд ROL, 1

и эффект: операнд = операнд, повернутый влево на одну позицию

или операнд ROL, CL

соответственно операнд = операнд, повернутый влево с (CL) поз.

Пример 1 (изначально AX = YYYY):

ROL AX, 1

Эффект:

1010 1010 1010 1010 B = AAAA H (rot. stg. cu 1)

-------------------------------------------------------------------

0101 0101 0101 0101 B = 5 5 5 5 H

Значение после выполнения инструкции: AX = 5555H, CF = 1.

Пример 2 (изначально AX = YYYY и CL = 4):

ROL AX, CL

Эффект:

1010 1010 1010 1010 B = AAAA H (rot. stg. cu CL)

----------------------------------------------------------------------

1010 1010 1010 1010 B = AAAA H

Значение после выполнения инструкции: AX = AAAAH, CF = 0.

2. Команды левого вращения с CF имеют вид:

RCL-операнд, 1

и эффект: операнд = (операнд, CF) повернут влево с о поз.

или операнд RCL, CL

соответственно операнд = (операнд, CF) повернут влево. с (CL) поз.

Пример 1 (изначально AX = YYYY и CF = 1):

RCL AX, 1

Эффект:

(CF = 1) 1010 1010 1010 1010 B = AAAA H (rot. stg. cu 1)

--------------------------------------------------------------------

(CF = 1) 0101 0101 0101 0101 B = 5 5 5 5 H

Значение после выполнения инструкции: AX = 5555H, CF = 1.

Пример 2 (изначально AX = YYYY и CF = 0):

RCL AX, 1

(CF = 0) 1010 1010 1010 1010 B = AAAA H (rot. stg. cu 1)

--------------------------------------------------------------------

(CF = 1) 0101 0101 0101 0100 B = 5 5 5 4 H

Значение после выполнения инструкции: AX = 5554H, CF = 1.

Пример 3 (изначально AX = YYYY, CL = 4 и CF = 1):

RCL AX, CL

Эффект:

(CF = 1) 1010 1010 1010 1010 B = AAAA H (rot. stg. cu CL)

----------------------------------------------------------------

(CF = 0) 1010 1010 1010 1101 B = AAAD H

Значение после выполнения инструкции: AX = AAADH, CF = 0.

Пример 4 (изначально AX = YYYY, CL = 4 и CF = 0):

RCL AX, CL

Эффект:

(CF = 0) 1010 1010 1010 1010 B = AAAA H (rot. stg. cu CL)

----------------------------------------------------------------------

(CF = 0) 1010 1010 1010 0101 B = AAA5 H

Значение после выполнения инструкции: AX = AAA5H, CF = 0.

3. Инструкции вращения вправо без CF имеют вид:

Операнд ROR, 1

и эффект: операнд = операнд, повернутый вправо на одну позицию

или операнд ROR, CL

соответственно операнд = операнд, повернутый вправо с (CL) поз.

Пример 1 (изначально AX = YYYY):

ROR AX, 1

Эффект:

1010 1010 1010 1010 B = AAAA H (rot. dr. cu 1)

-------------------------------------------------------------------

0101 0101 0101 0101 B = 5 5 5 5 H

Значение после выполнения инструкции: AX = 5555H, CF = 0.

Пример 2 (изначально AX = YYYY и CL = 4):

ROR AX, CL

Эффект:

1010 1010 1010 1010 B = AAAA H (rot. dr. cu CL)

-------------------------------------------------------------------

1010 1010 1010 1010 B = AAAA H

Значение после выполнения инструкции: AX = AAAAH, CF = 1.

4. Инструкции вращения вправо с CF имеют вид:

RCR операнд, 1

и эффект: операнд = (операнд, CF) повернут вправо с поз.

или операнд RCR, CL

соответственно операнд = (операнд, CF) повернут вправо с (CL) поз.

Пример 1 (изначально AX = YYYY и CF = 1):

RCR AX, 1

Эффект:

1010 1010 1010 1010 (CF = 1) B = AAAA H (rot. dr. cu 1)

------------------------------------------------------------------------

1101 0101 0101 0101 (CF = 0) B = D 5 5 5 H

Значение после выполнения инструкции: AX = D555H, CF = 0.

Пример 2 (изначально AX = YYYY и CF = 0):

RCR AX, 1

Эффект:

1010 1010 1010 1010 (CF = 0) B = AAAA H (rot. dr. cu 1)

------------------------------------------------------------------------

0101 0101 0101 0101 (CF = 0) B = 5 5 5 5 H

Значение после выполнения инструкции: AX = 5555H, CF = 0

Пример 3 (изначально AX = YYYY, CL = 4 и CF = 1):

RCR AX, CL

Эффект:

1010 1010 1010 1010 (CF = 1) B = AAAA H (rot. dr. cu CL)

---------------------------------------------------------------------------

0101 1010 1010 1101 (CF = 1) B = 5AAA H

Значение после выполнения инструкции: AX = 5AAAH, CF = 1.

Пример 4 (изначально AX = YYYY, CL = 4 и CF = 0):

RCR AX, CL

Эффект:

1010 1010 1010 1010 (CF = 0) B = AAAA H (rot. dr. cu CL)

--------------------------------------------------------

1010 1010 1010 0101 (CF = 1) B = AAAA H

Значение после выполнения инструкции: AX = AAAAH, CF = 1.

**7. Примеры программ**

***7.1. Расчеты с двойной точностью***

Основная роль инструкций накапливания с передачей и вычитание с остатком состоит в том, чтобы позволить вычислениям выполняться с двойной точностью (32 бита).

1. Сложение двух 32-битных значений - пар регистров (AX, BX) и (CX, DX) соответственно:

Изначально: AX = 7FFFH, BX = 8000H, CX = 4000H и DX = 0C000H.

ADD BX, DX; сборка LSW

ADC AX, CX ; сборка MSW

Эффект от последовательности:

AX BX +

CX DX

------------

AX BX

То есть:

0111 1111 1111 1111 1000 0000 0000 0000 B = 7FFF 8000 H +

0100 0000 0000 0000 1100 0000 0000 0000 B = 4000 C000 H

 (CF = 1)

--------------------------------------------------------------------------------------------

(CF = 0) 1100 0000 0000 0000 0100 0000 0000 0000 B = C000 4000 H

После выполнения последовательности: AX = C000H, BX = 4000H, CF = 0 (результат C000 4000 H).

2. Вычитание двух 32-битных значений - пары регистров (AX, BX) и (CX, DX) соответственно:

По умолчанию: AX = 7FFFH, BX = 8000H, CX = 4000H и DX = 0C000H.

SUB BX, DX; уменьшить LSW

SBB AX, CX; уменьшить MSW

Эффект от последовательности:

AX BX –

CX DX

------------

AX BX

То есть:

0111 1111 1111 1111 1000 0000 0000 0000 B = 7FFF 8000 H -

0100 0000 0000 0000 1100 0000 0000 0000 B = 4000 C000 H

(CF = 1)

--------------------------------------------------------------------------------------------

(CF = 0) 0011 1111 1111 1110 1100 0000 0000 0000 B = 3FFE C000 H

Значения после выполнения инструкции: AX = 3FFEH, BX = C000H, CF = 0 (результат 3FFE C000 H).

3. Вычисление суммы с двойной точностью переменных словарного типа a и b, результат помещается в переменную c.

СЕГМЕНТ ДАННЫХ

DAT SEGMENT

a dw 0a46fh

b dw 0dc89h

c dw ?, ?

DAT ENDS

COD SEGMENT

START:

MOV AX, DAT

MOV DS, AX

mov ax, a ; инициализации

mov dx, 0

add ax, b ; сумма с двойной точностью

adc dx, 0

mov c, ax ; результат запоминания

mov [c+2], dx

MOV AH, 4CH

INT 21H

COD ENDS

END START

***7.2. Расчеты с двойной точностью по движению***

Основная роль бита передачи в инструкциях перемещения и вращения - позволяет вычисления с двойной точностью (32-битные).

1. Смещение на позицию влево от значения 32 бита - пара регистров (AX, BX):

Изначально: AX = AAAAH и BX = 8888H.

SHL AX, 1; Смещение MSW

SHL BX, 1; Смещение LSW

ADC AX, 0; накапливание битов CF на LSW

Эффект от последовательности:

1010 1010 1010 1010 1000 1000 1000 1000 B = AAAA 8 8 8 8 H

/ / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /

---------------------------------------------------------------------------------------------------------

(CF = 1)0101 0101 0101 0101 0001 0001 0001 0000 B = 5 5 5 5 1 1 1 0 H

После выполнения последовательности: AX = 5555H, BX = 1110H, CF = 1 (результат 1 5555 1110H)

2. Смещение на позицию справа от значения 32 бита - пара регистров (AX, BX):

Начальное: AX = AAAAH и BX = 8888H.

SHR BX, 1; Смещение LSW

SHR AX, 1; Вытеснение ТБО

ADC DX, 0; накапливание битов CF на DX

MOV CL, 15; Подготовка 15-битного смещения

SHL DX, CL; CF сдвиг в позицию MSb

ADD BX, DX; накапливание битов CF на MSW

Эффект от последовательности:

1010 1010 1010 1010 1000 1000 1000 1000 B = AAAA 8 8 8 8 H

\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \

---------------------------------------------------------------------------------------------------------

0101 0101 0101 0101 0100 0100 0100 0100 (CF = 0) B = 5 5 5 5 4 4 4 4 H

После выполнения последовательности: AX = 5555H, BX = 4444H, CF = 0 (результат 5555 4444H).

3. Вращение с позицией влево от значения 32 бита - пара регистров (AX, BX):

По умолчанию: AX = AAAAH, BX = 8888H, DX = 0000H, CL = 4 и CF = 1.

ROL AX, 1; вращение MSW

RCL BX, 1; вращение (LSW, CF)

ADC AX, 0; накапливание битов CF на MSW

; получается вращением BX

Эффект от последовательности:

1010 1010 1010 1010 1000 1000 1000 1000 B = AAAA 8 8 8 8 H

/ / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /

---------------------------------------------------------------------------------------------------------

(CF = 1)0101 0101 0101 0101 0001 0001 0001 0001 B = 5 5 5 5 1 1 1 1 H

После выполнения последовательности: AX = 5555H, BX = 1111H, CF = 1 (результат 1 5555 1111H).

4. Вращение с позицией вправо от значения 32 бита - пара регистров (AX, BX):

По умолчанию: AX = AAAAH, BX = 8888H, DX = 0000H, CL = 4 и CF = 1.

ROR AX, 1; вращение верхнего слова (MSW)

RCR BX, 1; вращение младшего слова (LSW) с CF.

; так как вращение AX

ADC DX, 0; приведение CF в DX с момента вращения BX

ROR DX, 1; размещение в DX CF, полученного вращением

; BX, по наиболее значимой позиции

ADDAX, DX; накапливание до верхнего слова бита

; CF полученный вращением BX

Эффект от последовательности:

1010 1010 1010 1010 1000 1000 1000 1000 B = AAAA 8 8 8 8 H

\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \

---------------------------------------------------------------------------------------------------------

0101 0101 0101 0101 (CF = ) 0100 0100 0100 0100 (CF = 0) B = 5 5 5 5 4 4 4 4 H

После выполнения последовательности: AX = 5555H, BX = 4444H, CF = 0 (результат 5555 4444H).

5. Программа для перемещения с двойной точностью содержимого переменной v с двумя позициями вправо.

data segment

v dw 0abcdh, 0ef12h

data ends

cod segment

st:

mov ax, data

mov ds, ax

mov ax, x ; инициализации

mov bp, offset v

mov ax, ds:[bp]

mov bx, ds:[bp+2]

mov dx, 0 ; первое смещение

shr ax, 1

shr bx, 1

adc dx, 0

ror dx, 1

adc ax, 0

mov dx, 0 ; второе смещение

shr ax, 1

shr bx, 1

adc dx, 0

ror dx, 1

adc ax, 0

mov ds:[bp], ax ; результат хранения

mov ds:[bp+2], bx

mov ah, 4ch

int 21h

cod ends

end st

**8. Другие программы**

1. Программа для преобразования строчной буквы, прочитанной с клавиатуры, в прописную букву, отображаемую на экране.

data segment

numeprog db 25 dup(0ah),' Преобразование символа $'

citire db 2 dup(0ah),0dh,' Введите строчную букву: $'

afisare db 2 dup(0ah),0dh,' Прописная буква: $'

data ends

cod segment

start:

mov ax, data

mov ds, ax

mov dx, offset numeprog ; отображение строки символов

mov ah, 9 ; (название программы)

int 21h

mov dx, offset citire ; отображение строки символов

mov ah, 9 ; чтение сообщения

int 21h

mov ah, 1 ; чтение символов повторяется

int 21h ; экран(строчная буква)

sub al, 20h ; преобразование строчная буква -> прописная

 буква

mov bl, al

mov dx, offset afisare ; отображение строки символов

mov ah, 9 ; (отображение сообщения)

int 21h

mov dl, bl

mov ah, 2 ; отображение символов

int 21h ; (прописная буква)

mov ah, 8 ; чтение символов повторяется

int 21h ; экран (Enter)

mov ah, 4ch ; выход

int 21h

cod ends

end start

2. Программа для преобразования прописной буквы, прочитанной с клавиатуры, в строчную букву, отображаемую на экране.

data segment

numeprog db 25 dup(0ah),'Преобразование символа $'

citire db 2 dup(0ah),0dh,' Введите прописную букву $'

afisare db 2 dup(0ah),0dh,' Строчная буква: $'

data ends

cod segment

start:

mov ax, data

mov ds, ax

mov dx, offset numeprog ; отображение строки символов

mov ah, 9 ; (название программы)

int 21h

mov dx, offset citire ; отображение строки символов

mov ah, 9 ; (чтение сообщения)

int 21h

mov ah, 1 ; чтение символов повторяется

int 21h ; экран (прописная буква)

add al, 20h ; преобразование прописная буква -> строчная

 буква

mov bl, al

mov dx, offset afisare ; отображение строки символов

mov ah, 9 ; (отображение сообщения)

int 21h

mov dl, bl

mov ah, 2 ; отображение символов

int 21h ; (строчная буква)

mov ah, 8 ; чтение символов повторяется

int 21h ; экран (Enter)

mov ah, 4ch ; выход

int 21h

cod ends

end start

**9. Домашнее задание и упражнения.**

1. Напишите программу, которая использует исчисление двойной точности для вычисления суммы два слова, расположенные в регистрах AX и DX, с использованием только байтовых регистров (AH, AL, BH, BL, CH, CL, DH, DL).

2. Напишите программу, которая превращает символы 'a', ..., 'f', считываемые с клавиатуры, в значения 10, ..., 15.

3. Напишите программу, которая превращает символы "A", ..., "F", считываемые с клавиатуры, в значения 10, ..., 15.

4. Напишите программу, которая считывает с клавиатуры значения 10, ..., 15 (две цифры

последовательно), чтобы превратить их в символы 'A', ..., 'F' и отобразить их на экране.