МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. И. С. ТУРГЕНЕВА»

Кафедра «Информационные технологии»

**ОТЧЕТ**

по лабораторной работе № 1

на тему: «Линейное исполнение программ. Арифметические

и поразрядные логические операции над целыми двоичными числами»

по дисциплине: «Архитектура ЭВМ и систем»

Выполнил Шорин В.Д. Шифр:171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

Проверила: Конюхова О.В.

Отметка о зачете:

Дата: « » сентября 2017г.

Орёл.2017

Цель работы:

1.Изучение принципов функционирования памяти и микропроцессора компьютера при последовательном исполнении команд программы.

2. Приобретение навыков использования арифметических команд

при написании ассемблерных программ.

3. Приобретение навыков использования поразрядных логических команд при написании ассемблерных программ

4. Получение представления об особенностях обработки данных

разных размерностей и режимах доступа к данным при выполнении

арифметических и поразрядных логических операций.

Ответы к контрольным вопросам:

1

Сегмент – это прямоугольная область памяти, характеризующаяся начальным адресом и длиной.

Начальный адрес (адрес начала сегмента) – это номер (адрес)ячейки памяти, с которой начинается сегмент.

Длина сегмента – это количество входящих в него ячеек памяти. Сегменты могут иметь различную длину. Все ячейки, расположенные внутри сегмента, перенумеровываются, начиная с нуля. Адресация ячеек внутри сегмента ведется относительно начала сегмента; адрес ячейки в сегменте называется смещением или эффективным адресом - EA (относительно начального адреса сегмента).

2

Память компьютера разбита на следующие сегменты: сегмент данных для хранения операндов, сегмент кода для хранения операторов программы и сегмент стека – дополнительную память для временного размещения информации.

Начальные адреса сегментов помещаются микропроцессором в соответствующие сегментные регистры:

- Сегмент кода CS . Содержит команды программы.

- Сегмент данных DS. Содержит обрабатываемые программой данные.

-Сегмент стека SS. Этот сегмент представляет собой область памяти, называемую стеком. Работу со стеком процессор организует по следующему принципу: последний записанный в эту область элемент выбирается первым.

-Дополнительный (расширенный) сегмент данных ES. Используется, если программе недостаточно одного сегмента данных.

3

При выполнении арифметических операций используются 4 регистра общего назначения:

-AX/AH/AL (Accumulator register) – аккумулятор. Применяется

для хранения промежуточных данных;

– BX/BH/BL (Base register) – базовый регистр. Применяется для

хранения базового адреса некоторого объекта в памяти;

-CX/CH/CL (Count register) - регистр-счетчик. Применяется в командах, производящих некоторые повторяющиеся действия;

- DH/DL (Data register) - регистр данных. Так же, как и регистр AX/AH/AL, он хранит промежуточные данные;

4

Арифметические команды воздействуют на следующие флаги:

- CF (Carry Flag) – флаг переноса. Устанавливается в 1, если при выполнении арифметических и некоторых операций сдвига возникает «перенос» из старшего разряда;

- PF (Parity Flag) – флаг чѐтности. Проверяет младшие 8 битов результатов над данными. Чѐтное число единиц приводит к установке этого флага в 1, нечѐтное – в 0;

-AF (Auxiliary Carry Flag) – флаг логического переноса в двоично-десятичной арифметике. Устанавливается в 1, если арифметическая операция приводит к переносу или займу четвѐртого справа бита однобайтового операнда. Используется при арифметических операциях над двоично-десятичными кодами и кодами ASCII;

-ZF (Zero Flag) – флаг нуля. Устанавливается в 1, если результат операции равен 0, в противном случае ZF обнуляется;

-SF (Sign Flag) – флаг знака. Устанавливается в 1, если результат арифметической операции является отрицательным, в 0, если результат положительный;

-OF (Overflow Flag) – флаг переполнения. Устанавливается в единицу при арифметическом переполнении, когда результат выходит за пределы разрядной сетки.

5

Команды сложения и вычитания применяются для выполнения арифметических операций над целыми двоичными числами. Особенностью данных команд является то, что они записывают результат на место первого операнда (приѐмника), вследствие чего операнд – приѐмник должен находиться в регистре процессора. Операнд – источник может иметь любой режим адресации.

6

Особенностью операций умножения и деления является наличие в команде всего одного операнда (сомножителя или делителя), который может быть задан с помощью любого режима адресации данных, кроме непосредственного (находиться в памяти или в регистре). Второй операнд задан неявно; его местоположение фиксировано и зависит от размера операндов. Знаки результатов в операциях со знаком определяются по алгебраическим правилам. Если результат по размеру совпадает с размером сомножителей, то флаги CF и OF после завершения операции раны нулю, в противном случае - устанавливаются в единицу. Это значит, что результат

вышел за пределы младшей части произведения и состоит из двух частей, что необходимо учитывать при дальнейшей работе. Остальные флаги не определены.

Расположение операндов при умножении:

|  |  |  |
| --- | --- | --- |
| Первый  сомножитель | Второй  сомножитель | Результат |
| Байт | AL | 16 битов в AX; AL – младшая часть результата, AH – старшая часть результата  (AX)  (SRC) \* (AL) |
| Слово | AX | 32 бита в паре DX:AX; AX – младшая часть результата, DX – старшая часть результата  (DX:AX)  (SRC) \* (AX) |

Делитель может находиться в регистре или в памяти и иметь размер 8, или 16 битов. Местонахождение делимого фиксировано: если делитель имеет размер 1 байт, то делимое должно помещаться в регистр AX; если делитель является двухбайтовым, то старшие два байта делимого должны размещаться в регистре DX, а младшие два байта – в регистре AX. Результатом команды деления являются частное и остаток от деления, которые также размещаются в строго определѐнных регистрах

Расположение операндов при делении:

|  |  |  |  |
| --- | --- | --- | --- |
| Делимое | Делитель | Частное | Остаток |
| Слово (16 бит) в  регистре AX | Байт | Байт в регистре AL  (AL) (AX)/ (SRC) | Байт в регистре AH  (AH) (AX)/ (SRC) |
| Двойное слово (32 бита), в DX – старшая часть в AX – младшая часть | Слово | Слово в регистре AX  (AX)   (DX:AX)/ (SRC) | Слово в регистре DX  (DX)   (DX:AX)/(SRC) |

7

Логические команды выполняют логические операции над битами операндов. Размерность операндов должна быть одинакова. Логические команды наиболее часто используются для селективных (выборочных) установок, инвертирования, сброса или проверки битов в операнде – получателе в соответствии с двоичным значением операнда – источника. Такие действия часто встречаются в операциях над битами регистров и данных ввода-вывода.

|  |  |  |
| --- | --- | --- |
| Название команды | Мнемоника и формат команды | Описание действия |
| Инвертировать | NOT OPR | (OPR)not OPR  Инвертирование всех битов операнда |
| Объединить по «ИЛИ» | OR DST,SRC | (DST)(DST) or (SRC)  Выполнение операции логического «ИЛИ» над соответствующими парами битов источника и приѐмника |
| Объединить по «И» | AND DST,SRC | (DST)(DST) and (SRC)  Выполнение операции логического «И» над соответствующими парами битов источника и приѐмника |
| Сложить по MOD2 («исключающее ИЛИ») | XOR DST,SRC | (DST)(DST) xor (SRC)  Выполнение операции логического «Исключающее  ИЛИ» над соответствующими парами битов источника и приѐмника |
| Проверить | TEST OPR1,OPR2 | OPR1 and OPR2  Выполнение операции логического сравнения над соответствующими парами битов двух операндов |

8

Для установки разрядов маски в числе используется операция "битовое или" (OR) для числа и маски. Для сброса разрядов маски в числе используется операция "битовое и" (AND) для числа и инвертированной маски: число AND NOT маска.

9

Логические команды выполняют операции над битами операндов. Размерность операндов, естественно, должна быть одинакова. Например, если размерность операндов равна слову (16 бит), то логическая операция выполняется сначала над нулевыми битами операндов и ее результат записывается на место бита 0 результата. Далее команда последовательно повторяет эти действия над всеми битами с первого до пятнадцатого

Программа:

d\_s segment адрес начала сегмента данных

arg1 db 10 запишем число 10 по адресу аrg1

arg2 db 27 запишем число 27 по адресу arg2

arg3 db 10001000b запишем двоичное значение «10001000 » по адресу аrg3

sum db 0 в сумму помещаем 0

razn db 0 1 разность = 0

raznneg db 0 2 разность = 0

umn dw 1 1 произведение чисел = 1

umn2 dw 1 2 произведение чисел = 1

chastn db ? частное

ostatok db ? остаток от делении

d\_s ends адрес конца сегмента данных

assume ds:d\_s,cs:c\_s определяем каким регистрам соответствуют сегменты

c\_s segment адрес начала сегмента кода

begin: начало программы

mov ax, d\_s пересылка адреса начала сегмента данных в рабочий регистр ax

mov ds, ax пересылка регистра ax в ds

mov ah, arg1 пересылка в регистр ah значение arg1

mov al, arg2 пересылка в регистр al значение arg2

add ah,al сложение содержимого ah и al

mov sum,ah пересылка из ah в sum

mov ah,arg1 в регистр ah записываем значение аrg1

mov bh,arg2 в регистр bh записываем значение аrg2

sub ah,bh выполняем вычитание аrg1-arg2 (10-27)

mov razn,ah полученный результат записываем в ячейку razn

mov ah,arg1 в регистр ah записываем значение аrg1

mov bh,arg2 в регистр bh записываем значение аrg2

neg bh меняем знак arg2 (-27 получается)

sub ah,bh выполняем вычитание из arg1 arg2(10-(-27)

mov raznneg,ah полученный результат записываем в ячейку raznneg

mov al,arg1 в регистр al записываем значение аrg1

mov ah,arg2 в регистр ah записываем значение аrg2

imul ah умножить с учетом знака

mov umn,ax полученный результат записываем в ячейку umn

mov al,arg1 в регистр a записываем значение аrg1

mov ah,arg2 в регистр ah записываем значение аrg2

mul ah умножить без учета знака

mov umn2,ax полученный результат записываем в ячейку umn2

mov ah,arg2 в регистр ah записываем значение аrg2

mov bh,arg1 в регистр bh записываем значение аrg1

div bh делим arg2 на arg1

mov chastn,al помещаем частное в ячейку chastn

mov ostatok,ah помещаем остаток в ячейку ostatok

mov ah, arg3 в регистр ah записываем значение аrg3

or al,10001011b устанавливаем 2 последних бита в единицу

not al инвертируем число

and al,10001111b сбрасываем 3 бита

mov bh,al дублируем полученный результат в bh

xor bh,al складываем полученные значения по модулю 2

mov ah, 4ch завершаем программу, пересылаем в регистр ah

int 21h вызов прерывания 21h

c\_s ends конец сегмента кода

end begin конец программы