МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІЕХНІКА"

ІНСТИТУТ ПІСЛЯДИПЛОМНОЇ ОСВІТИ



**Реферат**

*з дисципліни*

***«Програмування процесорних систем»***

*на тему:*

# «Мова Асемблера мікропроцесора 8086»

Виконав: студент ІПДО

спеціальності 7.05010301

«Програмне забезпечення систем»

групи ПЗС-31

Гринчук Т.А.

Прийняв:

доцент Сенів М.М.

Львів - 2015

# ****Зміст****

# **Вступ....................................................................................................................................................3**

# **1. Архітектура процесора...........................................................................................................3**

# **1.1. Регістри....................................................................................................................................3**

# **1.2. Шина........................................................................................................................................5**

# **1.3. Робота з пам'ятю.....................................................................................................................5**

# **1.4. Система команд.......................................................................................................................5**

# **1.5. Периферійні пристрої.............................................................................................................6**

# **1.6. Мікрокомп'ютери на основі і8086……………………………………………………….....6**

# **1.7. Технічні характеристики…………………………………………………………………....7**

# **2. Мова Асемблера......................................................................................................................8**

# **2.1. Загальна характеристика........................................................................................................8**

# **2.2. Елементи операторів..............................................................................................................9**

# **2.3. Сегменти і процедури...........................................................................................................11**

# **2.4. Команди передачі даних......................................................................................................12**

# **2.5. Команди вводу/виводу.........................................................................................................14**

# **2.6. Команди арифметичних операцій.......................................................................................15**

# **2.7. Команди логічних операцій.................................................................................................18**

# **2.8. Команди передачі управління.............................................................................................19**

# **Висновки...........................................................................................................................................20**

# **Список використаних джерел.........................................................................................................21**

# ****Вступ****

Intel 8086 (також відомий як iAPX86) — перший 16-бітний мікропроцесор компанії Intel, що розроблявся з весни 1976 року і був випущений 8 червня 1978. Процесор мав набір команд, який застосовується і в сучасних процесорах, саме від нього бере свій початок відома на сьогодні архітектура x86.

Основними конкурентами мікропроцесора i8086 були Motorola 68000, Zilog Z80, чипсети F-11 і J-11 сімейства PDP-11, MOS Technology 65C816. Деякою мірою, в області військових розробок, конкурентами були процесори-реалізації MIL-STD-1750A.

Аналогом мікропроцесора i8086 був процесор NEC V30 (на 5% продуктивніший за i8086 і при цьому повністю з ним сумісний). Радянським клоном був мікропроцесор К1810ВМ86, що входив в серію мікросхем К1810.

Проект 8086 був розпочатий в травні 1976 року, і спочатку задумувався як тимчасова заміна для амбітного проекту iAPX 432, розвиток якого загальмувався. Це була спроба, з одного боку, протистояти 16-ти і 32-бітовим процесорам інших виробників (таких як Motorola, Zilog і National Semiconductor), а з іншого — боротьби із загрозою від Zilog Z80 (розробленого командою під керівництвом Фредеріко Фаджін, котрий пішов з Інтел), що став дуже успішним. Перша версія архітектури 8086 (система команд, переривання, робота з пам'яттю і вводом/виводом) була розроблена з середини травня до середини серпня Стівеном Морзе. Потім команда розробників була збільшена до чотирьох чоловік, які представили два основних проектних документи — 8086 Architectural Specifications і 8086 Device Specifications. При розробці не використовувалося спеціалізованих CAD-програм, а діаграми були виконані з текстових символів. Використовувалися вже випробувані елементи мікроархітектури і фізичної реалізації, в основному від i8085.

Ринок 8-розрядних мікропроцесорів в кінці 1970-х був переповнений, і Intel залишає спроби на ньому закріпитися та випускає свій перший 16-бітний процесор. Процесор i8086 являє собою модернізований процесор i8080 і, хоча, розробники не ставили перед собою мету досягти повної сумісності на програмному рівні, більшість програм написаних для i8080 здатні виконуватися і на i8086 після перекомпіляції. Новий процесор несе у собі безліч змін, які дозволили значно (в 10 разів) збільшити продуктивність у порівнянні з попереднім поколінням процесорів компанії.

# ****Архітектура процесора****

* 1. **Регістри**

Всього в процесорі i8086 було 14 16-розрядних регістрів (рис. 1.):

* 4 регістри загального призначення(AX, BX, CX, DX),
* 2 індексні регістри (SI, DI),
* 2 вказівні регістри (BP, SP),
* 4 сегментні регістри (CS, SS, DS, ES),
* *програмний лічильник або показник команди (IP)*
* *регістр прапорців (FLAGS, включає 9 прапорів*).

При цьому регістри даних (AX, BX, CX, DX) допускали адресацію не лише цілих регістрів, але і їх молодшої половини (регістри AL, BL, CL, DL) і старшої половини (регістри AH, BH, CH, DH), що дозволяло використовувати не лише нове 16-розрядне ПЗ, але зберігало сумісність і зі старими програмами (правда, їх необхідно було, принаймні, перекомпілювати).

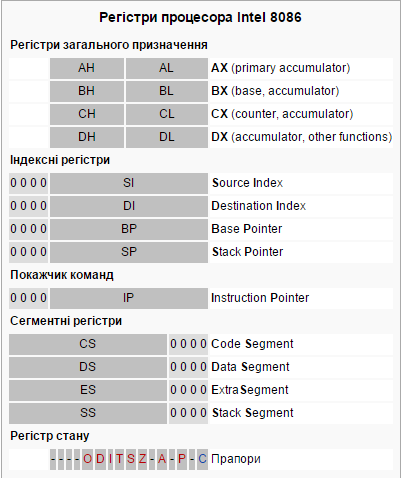


Рис. 1. Структура регістрів

***Лічильник команд, програмний лічильник, вказівник на поточну команду (англ. program counter, instruction pointer)*** — в архітектурі комп'ютера спеціалізований регістр процесора, за допомогою якого визначається, яка команда програми буде виконуватись процесором наступною (або яка поточна команда ним виконується - в залежності від прийнятого в тій чи інший архітектурі правила щодо завантаження лічильника).

Лічильник команд фактично зберігає адресу (номер комірки в пам'яті) цієї команди, а лічильником названий тому, що після (чи під час) виконання кожної команди збільшує своє значення на довжину виконаної команди, таким чином вказуючи на адресу, з якої потрібно завантажувати наступну команду до виконання. Деякі спеціальні команди, які можна умовно об'єднати в групу команд передачі управління (такі як команди умовного та безумовного переходу, виклику підпрограми, обробки переривання) примусово змінюють значення лічильника команд в залежності від певних умов, таким чином змінюючи послідовність виконання команд.

***Регістр стану (англ. Program State Word - PSW, англ. Flag Register - RF, англ. Condition Code Register - CCR)*** — це частина процесора, що зберігає важливу інформацію про стан обчислювальної системи, наприклад, біти-прапорці, які характеризують результати виконання арифметичних чи логічних операцій та порівнянь. Залежно від архітектурних особливостей, вміст регістра стану може бути частиною так званого контексту, що записується в стек при перериванні, або це покладено на плечі програміста.

Часто система команд передбачає спеціальні інструкції для читання та запису бітів регістра стану, оскільки вони застосовуються зі специфічною метою: для зміни природнього порядку слідування команд та управління процесором.

* 1. **Шина**

Розмір шини адреси був збільшений з 16 біт до 20 біт, що дозволило адресувати 1 Мбайт (220 байт) пам'яті. Шина даних була 16-розрядною. Проте в мікропроцесорі шина даних і шина адреси використовували одні й ті ж контакти на корпусі. Це призвело до того, що не можна одночасно подавати на системну шину адреси і дані. Мультиплексування адрес і даних в часі скорочує число контактів корпусу до 20, але й уповільнює швидкість передачі даних.

* 1. **Робота з пам'яттю**

Розмір шини адреси був збільшений з 16 біт до 20 біт, що дозволило адресувати 1 Мбайт (220 байт) пам'яті. Шина даних була 16-розрядною. Проте в мікропроцесорі шина даних і шина адреси використовували одні й ті ж контакти на корпусі. Це призвело до того, що не можна одночасно подавати на системну шину адреси і дані. Мультиплексування адрес і даних в часі скорочує число контактів корпусу до 20, але й уповільнює швидкість передачі даних.

* 1. **Система команд**

Система команд процесора i8086 складається з 98 команд (і більше 3800 їх варіацій): 19 команд передачі даних, 38 команд їх обробки, 24 команди переходу і 17 команд управління процесором. Можливі 7 режимів адресації. Мікропроцесор не містив команди для роботи з числами з плаваючою комою. Ця можливість реалізовувалася окремою мікросхемою, що називається математичний співпроцесор, який встановлювався на материнській платі. Співпроцесор зовсім не обов'язково мав бути зроблений Intel (модель i8087), наприклад, деякі виробники мікросхем, такі як Weitek, випускали продуктивніші співпроцесори, ніж Intel.

Система команд процесора i8086 включає в себе декілька дуже потужних рядкових інструкцій. Якщо інструкція має префікс REP (повтор), то процесор виконуватиме операції з блоками — переміщення блоку даних, порівняння блоків даних, присвоєння певного значення блоку даних певної величини, і так далі, тобто, одна інструкція 8086 з префіксом REP може виконувати 4-5 інструкцій, що виконуються на деяких інших процесорах. Але слід згадати, що подібні прийоми були реалізовані і в інших процесорах — Zilog Z80 мав інструкції переміщення і пошуку блоків, а Motorola 68000 може виконувати операції з блоками, використовуючи всього дві команди.

У мікропроцесорі i8086 була використана примітивна форма конвеєрної обробки. Блок інтерфейсу з шиною подавав потік команд до виконавчого пристрою через 6-байтову чергу команд. Таким чином, вибірка та виконання нових команд могли відбуватися одночасно. Це значно збільшувало пропускну спроможність процесора і позбавляло необхідності чекати зчитування команди з пам'яті при зайнятому іншими операціями інтерфейсі мікросхеми (у той час швидкість пам'яті значно випереджала швидкість цього процесора).

* 1. **Периферійні пристрої**
* Intel 8237 Контролер прямого доступу до пам'яті
* Intel 8251 USART
* Intel 8253 Програмований інтервальний таймер
* Intel 8255 Програмований периферійний інтерфейс
* Intel 8259 Програмований контролер переривань
* Intel 8279 Контролер клавіатури/дисплея
* Intel 8282/8283 8-Bit latch
* Intel 8284 Тактовий генератор
* Intel 8286/8287 Двонаправлений 8-бітний драйвер
* Intel 8288 Контролер шини
* Intel 8289 Арбітр шини
  1. **Мікрокомп'ютери на основі і8086**

У персональних комп'ютерах процесор i8086 практично не використовувався через дорожнечу спеціалізованих мікросхем, які були потрібні для роботи процесора. Це зрозуміли і в Intel, в 1979 році вона випускає процесор i8088, у якого шина даних була 8-бітовою.Через брак (ще не були розроблені) допоміжних 16-бітових мікросхем, і можливості використання великого парку 8-бітових, а також для здешевлення і зменшення розмірів плат, було вирішено випустити 8-бітовий варіант процесора (i8088). У 70-і роки мікросхеми динамічної оперативної пам'яті мали 1-бітову організацію і для 8 бітової системи було потрібно 8, а для 16-бітової — 16 мікросхем пам'яті. Тому випуск 8 розрядної версії здешевлював виробництво і зменшував розмір друкованої плати комп'ютера.

Але все ж в деяких мікрокомп'ютерах застосовувався і i8086, одним з таких є Mycron 2000 — перший комерційний мікрокомп'ютер на базі i8086. Машина для обробки текстів IBM Displaywriter, Compaq DeskPro і Wang Professional Computer також використовували i8086.

* 1. **Технічні характеристики**
* Дата анонсу: 8 червня 1978
* Тактова частота (МГц): від 4 до 10
* 5 (модель 8086), при частоті 4,77 продуктивність — 0,33 MIPS
* 8 (модель 8086-2, 0,66 MIPS)
* 10 (модель 8086-1, 0,75 MIPS)
* Приблизні витрати часу на операції, процесорних циклів (EA — час, необхідний для розрахунку ефективного адреси пам'яті, яке варіюється від 5 до 12 циклів):

+ Підсумовування: 3-4 (реєстрові), 9 + EA — 25 + EA — при операціях з пам'яттю + Множення: 70-118 (реєстрові), 76 + EA — 143 + EA — при операціях з пам'яттю + Переміщення даних: 2 (між регістрами), 8 + EA — 14 + EA — при операціях з пам'яттю

* Розрядність регістрів: 16 біт
* Розрядність шини даних: 16 біт
* Розрядність шини адреси: 20 біт
* Обсяг пам'яті, що адресується: 1 Мбайт
* Адресний простір I / O: 64 Кбайт
* Кількість транзисторів: 29 000
* Техпроцес (нм): 3000 (3 мкм)
* Площа кристала (кв. мм): ~ 30 (за іншими даними, 16 мм ²)
* Максимальна тепловиділення: 1,75 Вт
* Напруга живлення: +5 В
* Роз'єм: немає (мікросхема припаюється до плати)
* Корпус: 40-контактний керамічний чи пластиковий DIP, пізніше — 56-контактний QFP і 44-контактний PLCC
* Підтримувані технології: 98 інструкцій
* Обсяг черги команд: 6 байт

# ****Мова Асемблера****

* 1. **Загальна характеристика**

Мова асемблера являє собою машинну мову в символічній формі, яка більш зрозуміла і зручна людині.

Мова асемблера мікропроцесора Intel +8086 є досить складною, що в першу чергу пояснюється сегментної організацією пам'яті і одночасної адресацією чотирьох сегментів. У мові є понад 100 базових символічних команд, відповідно до яких асемблер генерує більше 3800 машинних команд. Крім того, в розпорядженні програміста є більше 20 директив, призначених для розподілу пам'яті, ініціалізації змінних, умовного асемблювання і т.д.

Вихідна програма на мові асемблера являє собою послідовність операторів. Оператори зазвичай займають один рядок. Асемблер сприймає оператори у вільному форматі, тобто елементам операторів не призначені фіксовані стовпці і між ними може бути будь-яке число прогалин там, де це необхідно.

Оператори у вихідній програмі класифікуються як командні оператори, оператори розподілу даних і директиви асемблера.

Командні оператори визначають генеруються асемблером машинні команди; вони містять мнемоніку і, при необхідності, один або два операнда. Кожен командний оператор породжує одну машинну команду, формат якої залежить від способу завдання операндів.

Оператори розподілу даних резервують комірки пам'яті для даних програми.

Директиви асемблера містять спеціальні вказівки для асемблювання програм і самі по собі не породжують ніяких машинних кодів.

**Формат командних операторів**

Командні оператори записуються в наступному форматі:

*Мітка: Префікс Мнемоніка операнд1, операнд2; Коментар*

де фігурні дужки позначають необов'язкові поля.

Розглянемо призначення окремих полів даного формату:

1) Мітка є визначуване користувачем ім'я, що закінчується двокрапкою. Значним мітки є поточне значення лічильника осередків (адрес) в поточному сегменті коду, тобто адреса зазначеної команди. Мітки як операнди використовуються тільки в командах передачі управління, і при цьому двокрапка наприкінці посилання на мітку не ставиться.

2) Префікс змушує асемблер сформувати один з префіксних байт - блокування LOCK або повторення REP, який безпосередньо передує команді.

3) Мнемоніка коду операції являє собою заздалегідь визначене та незмінний ім'я, яке ідентифікує тип генерується машинної команди.

4) Операнди задають адреси даних або самі дані, необхідні в даній команді.

5) Коментар призначений тільки для документування програми. Він завжди починається з крапки з комою.

**Формат директив асемблера**

Директиви асемблера і оператори розподілу даних мають дещо інший формат:

*Ім'я Директива Операнди; Коментар*

Розглянемо призначення окремих полів даного формату:

1) Ім'я директиви, на відміну від мітки, ніколи не закінчується двокрапкою. Деякі директиви вимагають обов'язкової наявності мітки.

2) Директива є одним з ключових незмінних слів асемблера і визначає його дії в процесі ассемблирования. Директиви використовуються програмістом для розподілу пам'яті, забезпечення зв'язку між програмними модулями та роботи з символічними іменами.

3) Операнди конкретизують дії, що виконуються за даною директиві.

4) Поле коментаря аналогічно такому ж полю в командних операторах.

* 1. **Елементи операторів**

Дамо визначення основних термінів, що будуть використані в подальшому.

Ключові (зарезервовані) слова являють собою імена, що мають для асемблера строго певний сенс. Їх не можна використовувати в якості ідентифікаторів.

Ідентифікатор як загальний термін для міток і імен змінних - це обумовлена ​​програмістом послідовність символів. Першим символом в послідовності повинна бути буква або один із символів @, підкреслення або знак питання. У якості подальших символів можна також використовувати цифри. Максимальна довжина ідентифікатора 31 символ.

Асемблер процесора 8086 є жорстко універсальна мова. Це означає, що операнди команд (регістри, змінні, мітки, константи) мають зв'язаний з ними атрибут типу, який повідомляє асемблеру деяку інформацію про операнд. Атрибут типу зазвичай мається на увазі за замовчуванням, але при необхідності може бути заданий явно.

**Регістри**

8-розрядних регістрів AL, AH, BL, BH, CL, CH, DL, DH приписаний тип BYTE, а 16-розрядних регістрів AX, BX, CX, DX, BP, SP, SI, DI і сегментних регістрів CS, DS, SS, ES приписаний тип WORD.

Розряди регістра прапорців являють собою однобітні регістри, для установки і скидання кожного з яких використовуються окремі машинні команди.

**Змінні**

При програмуванні може виникнути необхідність багаторазового звернення до даних. Замість того, щоб оперувати громіздкими чисельними значеннями адрес, зручно визначати і застосовувати символічні імена, відповідні адресами зазначених елементів.

Змінна - це одиниця програмних даних, що має символічне ім'я.

Більшість асемблерних програм починається з визначення даних, якими вони будуть оперувати. Розподіл елементів пам'яті і присвоєння їм ідентифікаторів здійснюється за допомогою директив DB (Define Byte - визначити байт), DW (Define Word - визначити слово), DD (Define Doubleword - визначити подвійне слово), DQ (Define Quadword - визначити 4 слова) або DT (Define Tenbyte - визначити 10 байтів).

Для завдання початкових значень можуть використовуватися числові константи і символьні ланцюжки. Якщо не потрібно задавати початкове значення змінної, то замість константи ставиться знак питання.

*Наприклад, оператор alpha DW 0Ah*

резервує слово пам'яті, присвоює йому ідентифікатор alpha і

заносить в нього код 000A;

*string DB 'Привіт'*

резервує 6 байт пам'яті і заносить в них рядок символів і

присвоює цьому рядку ідентифікатор string.

Щоб точно визначити тип змінної, на яку проводиться посилання, асемблер використовує оператори BYTE PTR, WORD PTR і DWORD PTR (покажчик на байт, слово і подвійне слово відповідно).

Для ініціалізації масивів застосовується конструкція DUP, яка в загальному випадку має вигляд:

*n DUP (поч. значення, поч. значення, ...)*

де параметр n задає число повторень елементів, що знаходяться в круглих дужках.

Наприклад, оператор *Addr DD 20 DUP (?)*

резервує місце для 20 повних адрес і присвоює цьому

масиву ідентифікатор Addr.

**Мітки**

Мітка є символічне ім'я для адреси комірки пам'яті і призначена для використання в якості операнда в командах управління.

**Числові константи**

Константа - це чисельне значення, що обчислюється під час асемблювання по заданому вислову.

Чисельні константи допускається представляти в системах числення з підставами 2, 8, 10 і 16. За молодшої цифрою повинен знаходитися однобуквений дескриптор системи числення: B - двійкова, O або Q - восьмерична, D (необов'язково) - десяткова, H-шістнадцятковий. Шістнадцяткова константа повинна бути доповнена зліва незначущим нулем.

**Символьні константи**

Символьна константа - це будь-який символ в коді ASCII. Символьний рядок може містити до 255 символів і повинна бути укладена в одиночні лапки.

* 1. **Сегменти і процедури**

Програми, написані на мові асемблера процесора 8086, можуть бути розділені на один або кілька сегментів. Кожен логічний сегмент має унікальне ім'я і однозначно відображається в сегментах пам'яті при завантаженні програми її виконання. Для визначення початкової і кінцевої осередків логічного сегмента в макроасемблері передбачені директиви SEGMENT (початок сегмента) і ENDS (кінець сегмента).

Зазвичай частина програми, що містить коди машинних команд,об'єднується в кодовий сегмент. Змінні, константи, таблиці та інші дані об'єднуються в сегмент даних. Для зберігання проміжних даних і при виклику підпрограм використовується сегмент стека. Іноді для зберігання даних може використовуватися четвертий, додатковий сегмент.

Директива ASSUME вказує асемблеру, до якому сегментному регістру належить той чи інший логічний сегмент. Якщо вся програма розміщується в одному сегменті пам'яті, то ця директива вказує, що всі сегментні регістри адресують один і той же сегмент.

Залежно від використовуваної моделі пам'яті в програмі можуть використовуватися один або кілька сегментів коду і один або декілька сегментів даних. Стековий сегмент як правило один.

Систему команд процесора 8086 утворюють 113 базових команд, багато з яких допускають використання різноманітних режимів адресації.

За функціональним призначенням виділяють спедующіе групи команд:

- Команди передачі даних;

- Команди арифметичних операцій;

- Команди логічних операцій і зрушень;

- Команди передачі управління;

- Ланцюгові команди;

- Команди управління мікропроцесором.

* 1. **Команди передачі даних**

Команди передачі даних поділяють на 4 підгрупи:

- Загальні команди передачі даних;

- Стекові команди;

- Команди введення-виведення;

- Команди передачі ланцюжків байт або слів.

Команди передачі даних не модифікують стану прапорців. Виняток становлять команди POPF і SAHF, прямо впливають на регістр прапорців.

Далі будуть використовуватися наступні позначення: dst - одержувач,

src - джерело,

mem - адреса пам'яті (зсув), заданий будь-яким методом адресації,

reg - регістр загального призначення,

sreg - сегментний регістр,

AС - регістр-акумулятор (AL або AX),

data - безпосередні дані.

ЗАГАЛЬНІ КОМАНДИ ПЕРЕДАЧІ ДАНИХ

У цю підгрупу входять команди, що здійснюють передачу регістр-регістр, регістр-пам'ять і пам'ять-регістр. Найбільш потужною серед них є команда MOV.

Команда MOV

Ця команда має наступне узагальнене уявлення:

MOV dst, src

тобто перший вказується операнд-одержувач, а другим - операндісточнік. Одним з операндів обов'язково повинен бути регістр.

Команда здійснює передачу: регістр - регістр, регістр - пам'ять, пам'ять - регістр, безпосередні дані - регістр, безпосередні дані - пам'ять, регістр - сегментний регістр, пам'ять - сегментний регістр, сегментний регістр - регістр, сегментний регістр - пам'ять.

Допустимі формати команди:

MOV mem / reg1, mem / reg2

MOV mem / reg, data

MOV sreg, mem / reg

MOV mem / reg, sreg

Команда обміну XCHG

Команда обміну XCHG дозволяє обмінювати вміст будь-якого загального регістра і комірки пам'яті, а також будь-якої пари загальних регістрів.

Формат команди:

XCHG reg, mem / reg

Команда XLAT

Команда перетворення XLAT застосовується для швидкого перетворення символів з одного коду в інший. Вона заміняє вміст акумулятора AL на байт з 256-байтовой таблиці, початкова адреса якої знаходиться в регістрі BX, а восьмибітового зсуву - в регістрі AL. В якості сегментного використовується регістр DS.

Формат команди:

XLAT

Команди LEA, LDS і LES

Команди LEA, LDS і LES призначені для завантаження ефективного адреси.

Команда LEA витягує з пам'яті 16-бітову адресу і поміщає його в один із загальних регістрів. Практично ця команда дублює один з варіантів команди MOV, але виконується швидше.

Команда LDS витягує з пам'яті 32-бітову адресу і поміщає перша лічену з пам'яті слово в заданий загальний регістр, а друге - в сегментний регістр DS.

Команда LES витягує з пам'яті 32-бітову адресу і поміщає перша лічену з пам'яті слово в заданий загальний регістр, а друге - в сегментний регістр ES.

Зазвичай в команді LDS вказується регістр SI, а в команді LES регістр DI.

Формат команд:

LEA reg, mem

LDS reg, mem

LES reg, mem

Команди LAHF і SAHF

Команда LAHF передає молодший байт регістра прапорців в регістр AH, а команда SAHF реалізує зворотний передачу - вміст регістра AH передається в молодший байт регістра прапорців (старший байт не змінюється).

Формат команд:

LAHF

SAHF

СТЕКОВІ КОМАНДИ

Кожна команда занесення даних в стек PUSH має відповідну їй команду вилучення з стека POP. Для адресації вершини стека використовується стековий покажчик SP. Всі стекові команди маніпулюють тільки словами і супроводжуються автоматичної модифікацією покажчика стека: при включенні в стек виробляється декремент, а при вилученні з стека - інкремент SP.

До виконання стекових команд регістри SP і SS повинні бути ініціалізованні.

Команда PUSH включає в стек вміст адресуемого регістра або комірки пам'яті, а команда POP витягує вміст вершини стека і передає його в регістр або елемент пам'яті.

Команди POSHF і POPF призначені для тимчасового запам'ятовування в стеку і подальшого відновлення зі стека вмісту регістра прапорів. З їх допомогою можна змінювати стан прапора трасування TF, оскільки команд прямого впливу на цей прапор немає.

Формат команд:

PUSH mem / reg POP mem / reg

PUSH sreg POP sreg

PUSHF POPF

* 1. **Команди вводу/виводу**

Команда введення IN і команда виведення OUT допускають роботу як з байтами, так і зі словами. Команда IN завантажує дані з заданого порту в акумулятор, а команда OUT виконує передачу з акумулятора в порт. Для портів вводу-виводу в діапазоні 00-FF можна використовувати пряму укорочену адресацію, а інші порти в діапазоні 100-FFFF можна адресувати тільки побічно через регістр DX.

Формат команд:

IN ac, port OUT port, ac (пряма укорочена адресація)

IN ac, DX OUT DX, ac (непряма адресація)

* 1. **Команди арифметичних операацій**

Процесор +8086 має широкий набір команд, що реалізують арифметичні операції, що дозволяє застосовувати його в складних системах обробки даних.

Арифметичні операції виконуються над цілими числами чотирьох типів: беззнаковими двійковими, знаковими двійковими, упакованими десятковими і неупакованими десятковими. Довжина чисел може бути 8 або 16 біт.

Діапазони чисел: беззнакове 8-бітове: 0 - 255, беззнаковое 16-бітове: 0 - 65535, знакове 8-бітове: -128 - +127, знакове 16-бітове: -32768 - +32767.

Команди арифметичних операцій впливають на стан прапорців OF, SF, ZF, AF, PF і CF.

Одні й ті ж команди додавання і віднімання можуть використовуватися для операцій як над беззнаковими, так і знаковими числами. Контроль над типами чисел повинен виконувати сам програміст.

Команда ADD

Команда ADD дозволяє виробляти додавання 8- або 16-бітових двійкових чисел в режимі регістр-регістр, регістр-пам'ять і пам'ять регістр, причому адресація пам'яті здійснюється в будь-якому допустимому режимі. Загальне уявлення команди має вигляд

ADD dst, src

тобто перший операнд складається з другим і результат операції

заміщає перший операнд.

Формат команди:

ADD mem / reg1, mem / reg2

ADD mem / reg, data

Команда ADC

Команда ADC виконує додавання з переносом: на відміну від команди ADD в операції додавання бере участь прапорець CF, значення якого додається до молодшого біту результату складання операндів.

Формат команди:

ADC mem / reg1, mem / reg2

ADC mem / reg, data

Команда INC

Команда INC дозволяє збільшити на 1 вміст будь-якого загального регістра або комірки пам'яті.

Формат команди:

INC mem / reg

КОМАНДИ віднімання

Команда SUB дозволяє виробляти віднімання 8- або 16-бітових двійкових чисел. Загальне уявлення команди має вигляд

SUB dst, src

тобто другий операнд віднімається з першого і результат операції заміщає перший операнд.

Формат команди:

SUB mem / reg1, mem / reg2

SUB mem / reg, data

Команда DEC

Команда DEC дозволяє зменшити на 1 вміст будь-якого загального регістра або комірки пам'яті.

Формат команди:

DEC mem / reg

Команда NEG

Команда NEG змінює знак числа, що знаходиться в регістрі або комірці пам'яті, на протилежний.

Формат команди:

NEG mem / reg

КОМАНДА ПОРІВНЯННЯ

Команда порівняння CMP виконує віднімання другого операнда з першого, але ніде не запам'ятовує результат операції і впливає тільки на стан прапорців.

Формат команди:

CMP mem / reg1, mem / reg2

CMP mem / reg, data

КОМАНДИ МНОЖЕННЯ

Мікропроцесор +8086 має дві команди множення: для беззнакових і для знакових двійкових чисел. Множення десяткових чисел вимагає використання спеціальних команд корекції, які будуть розглядатися пізніше.

Команда MUL

Команда множення беззнакових цілих чисел MUL виконує множення адресуемого операнда на вміст акумулятора. Загальне уявлення команди має вигляд

MUL src

При операції над байтами функції акумулятора виконує регістр AL, а 16-бітний результат операції міститься в регістр AX. При операції над словами функції акумулятора виконує регістр AX, а твір довжиною 32 біта формується в регістрах DX (старше слово) і AX (молодше слово).

Формат команди:

MUL reg

MUL mem

Команда IMUL

Команда IMUL аналогічна команді MUL, але співмножники і твір інтерпретуються як знакові двійкові числа в додатковому коді.

Формат команди:

IMUL reg

IMUL mem

КОМАНДИ ДІЛЕННЯ

Мікропроцесор +8086 має дві команди ділення: для беззнакових і для знакових двійкових чисел. Ділення десяткових чисел також вимагає використання спеціальних команд корекції.

Команда DIV

Команда ділення беззнакових чисел DIV робить розподіл вмісту акумулятора та його розширення на вміст адресуемого операнда.

При розподілі 16-бітного діленого на 8-бітний дільник ділене поміщають в регістр AX. В результаті виконання операції приватне формується в регістрі AL, а залишок - в AH.

При розподілі 32-бітного діленого на 16-бітний дільник старша частина діленого міститься в регістр DX, а молодша - в AX. В результаті виконання операції приватне формується в регістрі AX, а залишок - в DX.

При розподілі на 0 автоматично відбувається переривання і перехід до спеціальної програми обробки.

Формат команди:

DIV reg

DIV mem

Команда IDIV

Команда IDIV аналогічна команді DIV, але ділене, дільник і приватне інтерпретуються як знакові двійкові числа в додатковому коді.

Формат команди:

IDIV reg

IDIV mem

* 1. **Команди логічних операцій**

Логічні операції представлені командами NOT (інверсія), AND (кон'юнкція), OR (диз'юнкція), XOR (виключає АБО) і командою TEST, яка виконує кон'юнкцію операндів, але не змінює їх значень. Всі логічні операції є Поразрядно, тобто виконуються незалежно для всіх біт операндів.

Бінарні команди AND, OR, XOR і TEST впливають на прапорці OF, SF, ZF, PF і CF. Унарна операція NOT не впливає на стан прапорців.

Формати команд:

AND mem / reg1, mem / reg2

AND mem / reg, data

OR mem / reg1, mem / reg2

OR mem / reg, data

XOR mem / reg1, mem / reg2

XOR mem / reg, data

TEST mem / reg1, mem / reg2

TEST mem / reg, data

NOT mem / reg

* 1. **Команди передачі управління**

Сегментна організація програмної пам'яті визначає два основні різновиди команд передачі управління. Передача управління в межах поточного сегменту коду називається внутрисегментного - при цьому модифікується тільки регістр IP та адресу переходу може бути представлений одним словом. Така передача управління називається ближньої (тип NEAR), а її варіант з сокращеннним діапазоном адрес переходів - короткою. Передача управління за межі поточного сегмента коду називається межсегментной або далекої (тип FAR) - при цьому необхідно модифікувати вміст регістрів IP і CS і адреса переходу представляється двома словами (сегмент: зсув).

Команди передачі управління процесора 8086 підрозділяються на команди безумовних переходів, умовних переходів, викликів, повернень, управління циклами і команди переривань.

Команди передачі керування не змінює стан регістра прапорів (за винятком команди IRET).

КОМАНДИ безумовного переходу

Команди безумовного переходу мають спільну мнемоніку JMP. Команда короткого безумовного переходу містить у другому байті зсув, яке інтерпретується як знакове ціле. Діапазон значень байта зміщення становить -128 - +127. Якщо зсув позитивне, здійснюється перехід вперед, а якщо негативне - перехід назад.

Команда ближнього безумовного переходу може або безпосередньо містити 16-бітове зсув, або непрямий адреса 16-бітного зміщення. Діапазон зміщення становить -32768 - +32767 байт щодо адреси команди, що знаходиться після команди JMP.

Команда далекого безумовного переходу реалізує прямий і непрямий міжсегментного переходи.

Формати команд:

JMP dispL - короткий перехід

JMP disp - ближній прямий перехід

JMP mem / reg - ближній непрямий перехід

JMP addr - далекий прямий перехід

JMP mem - далекий непрямий перехід

**Висновки**

Процес трансляції програми на мові асемблера в об'єктний код прийнято називати асемблювання. На відміну від компілювання, асемблювання - більш-менш однозначний і оборотний процес. У мові асемблера кожної мнемоніці відповідає одна машинна інструкція, в той час як в мовах програмування високого рівня за кожним виразом може ховатися велика кількість різних інструкцій. В принципі, це поділ є досить умовним, тому іноді трансляцію асемблерних програм також називають компіляцією.

**Список використаних джерел**

1). Вострикова З. П. Программирование на языке ассемблера ЕС ЭВМ. М.: Наука, 1985.

2). Галисеев Г. В. Ассемблер для Win 32. Самоучитель. — М.: Диалектика, 2007. — С. 368. — ISBN 978-5-8459-1197-1.

3). Зубков С. В. Ассемблер для DOS, Windows и UNIX.

4). Ірвин Кип. Язык ассемблера для процессоров Intel = Assembly Language for Intel-Based Computers. — М.: Вильямс, 2005. — С. 912. — ISBN 0-13-091013-9.

5). Калашников О. А. Ассемблер? Это просто! Учимся программировать. — БХВ-Петербург, 2011. — С. 336. — ISBN 978-5-9775-0591-8.