کتاب مرجع   برنامه نویسی به زبان اسمبلی

دکتر حسین دلاوری انتشارات آیلار

بارم امتحان

۱۵ نمره امتحان پایانی تستی و تشریحی

۵ نمره کار در کلاسی و پروژه پایان ترم

شناخت ماشین

در این مفاهیم درسی علاوه بر شناخت ساختار و اصول اولیه زبان شناسی اسمبلی با ساختار داخلی و عملکرد درونی عناصر در ماشین آشنا خواهیم شد که مرتبط با مدیریت زبان اسمبلی خواهند بود به عبارتی ساده‌تر بخش های از ساختار ماشین را مورد بررسی قرار خواهیم داد که بتوانیم در مدیریت زبان ماشین و روال های کاری دسترسی داشته باشیم.

سوال

سیستم های عددی (مبناها) دارای چه مفاهیمی در یک ماشین می باشند.

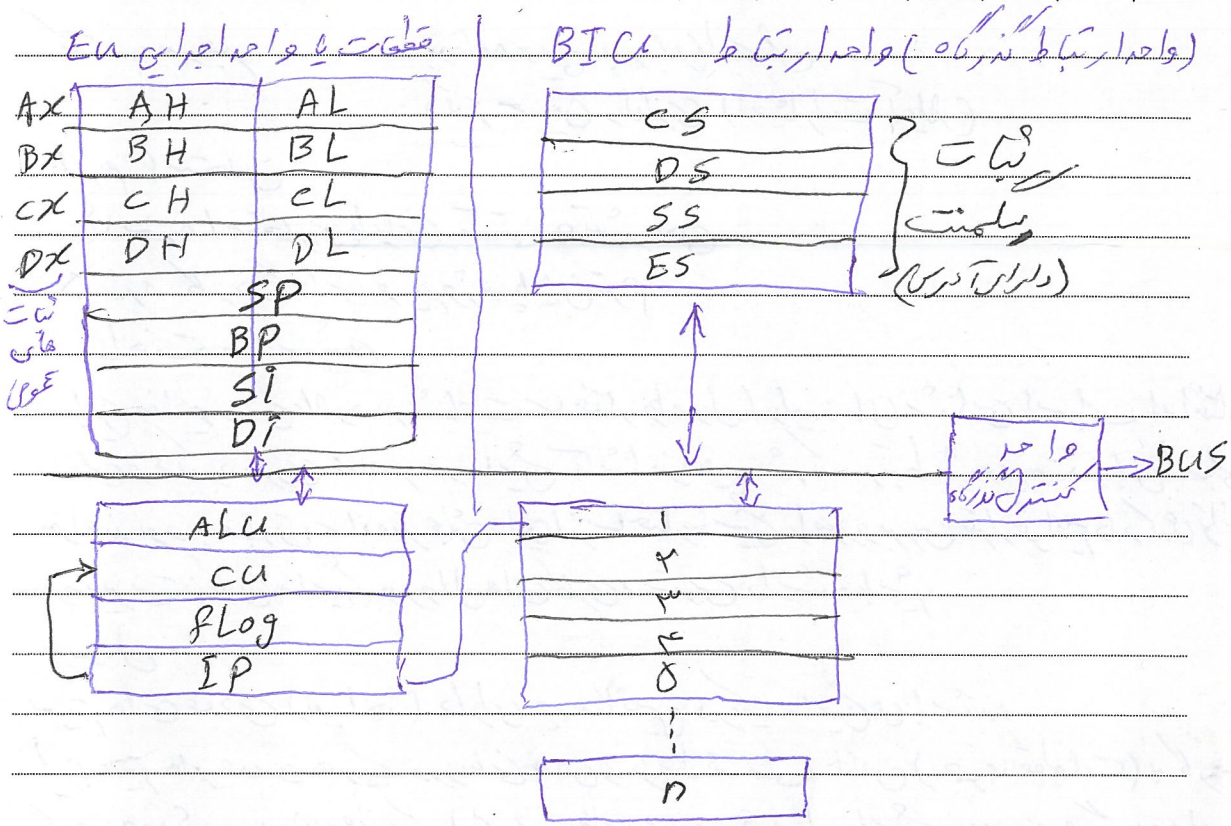
این سیستم های عددی هر یک در زمانی خاص و توسط قسمتی خاص (مجموعه قطعات) به کار گرفته می شوند بدین معنی که این سیستم های عددی توسط ماشین مدیریت می گردد. برای مثال در هنگام ذخیره سازی داده ها و ورودی ها از مبنای ۱۶ در هنگام محاسبات و پردازش (سیگنال های انتقال) مفهوم مبنای ۲ و همچنین در انتقال های جانبی (در برخی از مواقع) از مبنای ۸ استفاده می کنیم.

نکته ۱

در مفاهیم ماشین و برنامه نویسی اسمبلی یک سیستم وجود دارد که در آن باید تصور کنیم سخت افزار و نرم افزار ها باهم ترکیب و عجین شده است.

ساختار cpu

شکل زیر نمایش دهنده ساختار cpu می باشد که از طریق زبان ماشین می توان به آن دسترسی داشته و بر آن مدیریت نماییم. نکته حائز اهمیت این است که این شکل یک تصور منطقی است.



نمای منطقی CPU

شناسایی وظایف و قابلیت های جزئی در عناصر CPU

دیاگرام ترسیم شده (شکل فوق) یک تقسیم بندی منطقی از عناصری است که در اجرای فرامین و دستورات زبان ماشین دخالت دارند.

سوال

به نظر شما به جز عناصر ترسیم شده در شکل فوق عناصر دیگری در CPU فعالیت دارند؟ بله، عناصر و قطعات تخت افزاری دیگری نیز وجود دارند که هر یک دارای وظیفه ای می باشند اما از طریق ساختار زبان ماشین قابل دسترسی به طور عام نمی باشند

سوال

هر ثبات دارای چند دسته وظیفه یا ماموریت می باشد؟

حداقل ۲ دسته وظیفه و ماموریت است.

الف) ماموریتی که توسط ماشین تعیین می شود.

ب) ماموریتی که توسط برنامه نویس تعیین و استفاده می شود.

تذکر: ثبات در هنگام مدیریت ارجحیت را به ماشین می دهد.

سوال

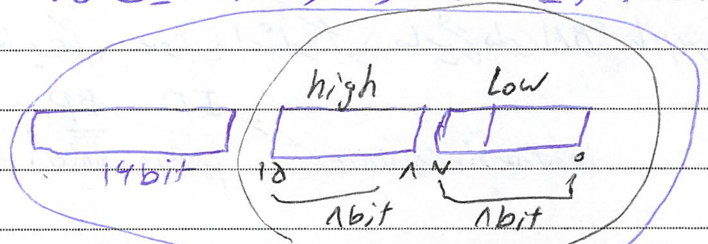
انواع ثبات ها را در cpu نام ببرید.

۱- ثبات های عمومی یا چند منظوره

کلیه ثبات ها با نام های AX، BX، CX و DX را می گویند. این ثبات ها از دو بخش تشکیل شده اند قسمت کم ارزش AL (L می باشد) و قسمت پر ارزش AH (H می باشد)

AL Low AH High

ثبات در ماشین های IBM با ظرفیت های ۸ و ۱۶ و ۳۲ و ۶۴ بیتی می باشند



External ← EAX ۳۲bit AX

AX

این ثبات می تواند در کلیه عملیات های محاسباتی نظیر جمع، ضرب، تفریق و تقسیم شرکت نموده و حتی برای نگهداری موقت داده ها ـ اما ماموریتی که از سوی ماشین بر روی این ثبات است شرکت کردن در عملیات ورود و خروج می باشد و استفاده از آن به عنوان تعیین سرویس

BX

این ثبات نیز مانند ثبات AX همه منظوره بوده که می توان از آن در انجام محاسبات استفاده نمائیم. اما وظیفه ای که ویژه آن است شرکت کردن به عنوان یک شاخص و اندیس در آدرس دهی می باشد

CX

این ثبات مانند دو ثبات قبلی می تواند در کلیه عملیات محاسباتی و عملیاتی به کار گرفته شود. اما ماموریت ویژه آن شرکت کردن در دستور loop به عنوان یک شمارنده می باشد به طوری که اگر بخواهیم در سطح زبان ماشین حلقه ای شبیه for ایجاد نماییم از ثبات CX استفاده می کنیم

DX

این ثبات در کلیه عملیات ضرب، تقسیم و محاسباتی دیگر شرکت کرده و به عنوان ثبات

تکمیل کننده، تعداد بیت‌های ممکن استفاده می‌شود، اما وظیفه خاصی از آن این است که از این بیت‌ها به عنوان تعیین آدرس و یا بیت‌های دوم اعداد در ضرب و تقسیم استفاده می‌شود.

۲- ثبات‌های ایندکس SI و DI

شامل SI و DI می‌باشند. این دو ثبات دارای ماهیتی یکسان برنامه‌نویسی می‌باشند. به طوری که در هر دو عملیات‌های آدرس‌دهی و تعیین نشانی استفاده می‌شوند. از آن‌ها می‌توانیم در پردازش رشته‌ها و داده‌های آرایه استفاده نماییم.

۳- ثبات‌های اشاره‌گر

ثبات‌هایی که به عنوان اشاره‌گر در ماشین‌های IBM نام‌برده می‌شود عبارت هستند از:

IP ، BP ، SP

instructer pointer ← basic pointer | stack pointer

(SP) stack pointe

این بیت حاوی آدرس آفست (offset) برای دسترسی به خانه‌های حافظه SP+SS → و یا stack segment می‌باشد. (بیتی از حافظه متفاوت در حافظه می‌باشد)

BP

این بیت برای آدرس‌دهی حافظه به کار گرفته می‌شود که می‌تواند به عنوان آفست برای stack segment و Extra segment (ES) نیز استفاده شود.

این بیت را می‌توانیم برای آدرس‌دهی پایه و غیره استفاده کنیم (برای انتقال BP+ES → یا رجیستر نیز استفاده می‌شود)

IP

این بیت حاوی آدرس آفست مرتبط با code segment و یا CS می‌باشد. این طریق دستورالعملی که قرار است توسط ماشین پردازش شود نشانی و به صف درون بافر جهت دستورالعمل اجرا استقبال می‌شود.

نکته

IP با آدرس درون CS ترکیب می‌شود. IP+CS →

CU (واحد کنترل)

این قطعه درون CPU با استفاده از الگوریتم های مختلف به شناسایی دستورالعمل های کد از روی حافظه RAM می پردازد و سپس فرمان لازم در جهت نحوه پردازش و چگونگی آن را به قسمت ALU و دیگر عناصر مرتبط برای اجرا صادر می نماید. (شناسایی تفسیر و انتخاب)

سوال

fetch یا واکشی چیست؟ دریافت داده یا دستورالعمل از حافظه RAM را می گویند که در ۲ مرحله انجام می گردد که با تعیین آدرس فیزیکی محل داده و دستورالعمل، داده و دستورالعمل های لازم به سمت CPU ارسال می گردد.

سوال

چرخه پردازشی به طور عام چگونه است؟

ابتدای امر مقدار ثبات IP با ثبات CS (code segment) جمع و محل مورد نیاز در حافظه RAM تعیین می گردد. سپس حافظه RAM براساس این آدرس داده و دستورالعمل لازم را به سمت CPU و صف منتقل می کند. در این زمان CU به شناسایی، تفسیر و انتخاب پرداخته و ALU به پردازش لازم خواهد پرداخت.

نتیجه پردازش برای شناسایی وضعیت کار انجام شده در ثبات Flag ذخیره می گردد به طوری که از طریق آن می توانیم مشکلات نتیجه پردازش را شناسایی نماییم. پس از پایان میزان ثبات IP تغییر نموده تا اینکه دستور بعدی را واکشی نماید.

سوال

سگمنت چیست؟ بخشی از حافظه RAM با ظرفیت 64KB می باشد که به سه نوع وجود دارد

۱- کد سگمنت CS

۲- دیتا سگمنت DS

۳- استک سگمنت SS

هر سگمنت دارای یک آدرس شروع می باشد که در ثبات سگمنت قرار می گیرد

آفست چیست

آدرس مکانی داخل هر سگمنت که از 0000 الی FFFF تغییر می یابد را می گویند.

تذکر

ثبات IP نشان دهنده آفست است

آدرس شروع هر سگمنت باید بر عدد 10H بخش پذیر باشد

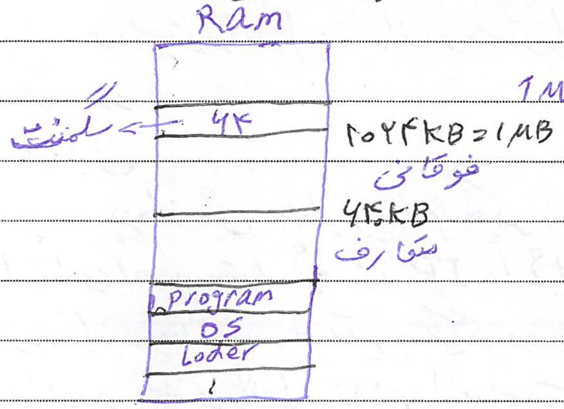
سوال

ساختار حافظه رم در ماشین IBM به صورت زیر است.

الف- بخش متعارف کمتر از 640k

ب- بخش فوقانی بین 640k و 1MB

ج- بخش بالاتر از 1MB



انواع سگمنت

به طور کلی ۳ نوع سگمنت مطرح می باشد که هر یک حاوی داده و اطلاعات مخصوص به خود می باشد. شبه دستور مربوط به سگمنت به صورت زیر می باشد.

برچسب segment para combine class

- class: به معنی این است که این سگمنت جزء کدام خانواده قرار دارد
- combine: نحوه ترکیب هر سگمنت نسبت به سگمنت های دیگر
- para: نحوه تقسیم بندی در حافظه تخصیصی

⋮

برچسب Ends

گونه های سگمنت مشخص و تعریف شده می باشند اما نحوه بهره گیری آنها و نحوه ارتباط آنها وابسته به مدیریت پروژه است که سگمنت ها را چگونه باهم ترکیب نماییم و همچنین از درون یک سگمنت می توان درون سگمنت های دیگر پرش نماییم برای مثال اگر ذهنیت به سمت تنها وجود ۱ سگمنت کلی در یک پروژه باشیم این حالت صحیح نیست.

Date: Subject: ۷

مفهوم combine در دستور segment چیست؟

بدین معنی است که ما خواهیم برحسب نیاز و ضرورت یک پروژه به تعداد دلخواه segment معرفی نموده و آنها را در کنار هم استفاده نماییم. که این ترکیب می تواند به صورت های مختلف انجام شود نکته:

کد اسکی اعداد ها الی ۹ بر مبنای ۱۶ عبارت از ۳۰ الی ۳۹ ۳۹ = ۹ ← ۳۰ + عدد

کد اسکی اعداد ۰ الی ۹ بر مبنای ۱۰ عبارت از ۴۸ الی ۵۷

کد اسکی حروف A الی Z بر مبنای ۱۶ عبارت از ۴۱ الی ۹۶ A = ۴۱

کد اسکی حروف A الی Z بر مبنای ۱۰ عبارت از ۶۵ الی ۹۰

کد اسکی حروف a الی z بر مبنای ۱۶ عبارت از ۶۱ الی ۸۶ a = ۶۱

کد اسکی حروف a الی z بر مبنای ۱۰ عبارت از ۹۷ الی ۱۲۲

مثال

برنامه ای بنویسید که دو عدد یک رقمی را از ورودی دریافت و پس حاصل جمع آن را نشان دهد.

data segment ← دریافت

f db 10

برروش جمع و تقسیم

Mov AH, 01 } دریافت عدد اول از ورودی
int 21H }

Mov BL, AL

int 21H

sub AL, 30H

sub BL, 30H

Mov BL, AL

Div AL, 10f

add AL, 30H

add AH, 30H

Mov BL, AH

Mov AH, 02H دریافت عدد دوم از ورودی

Raz

```
Mov DL, AL
int 21H
Mov AH, 02H
Mov DL, BL
int 21H
```

```
$db 10
Mov AH, 01
int 21H
Mov BL, AL
Mov AH, 01H
int 21H
sub AL, 30H
sub BL, 30H
add AL, BL
add AL, 0110B
Mov BL, AL
and AL, 00001111B
and BL, 11110000B
shR BL, 4
add AL, 30H
add BL, 30H
Mov AH, 02H
Mov DL, AL
int 21H
Mov AH, 02H
Mov DL, BL
int 21H
```

بررسی and و or و شیفت کردن

دریافت عدد از ورودی

شیفت به راست

Date: Subject: 9

ساختار برنامه
قالب

در زبان اسمبلی دو دو پایه می توانیم برنامه بنویسیم

قالب اول: ساختار EXE

قالب دوم: ساختار com

هر یک از این قالب ها دارای خصوصیات و ویژگی هایی می باشند. برای مثال در قالب com حجم و اندازه برنامه ها کوچک می باشد ولی انعطاف پذیری و مدیریت برنامه نویسی در EXE وجود ندارد.

قالب برنامه در حالت EXE

در این حالت شکل و الگوی زیر رعایت می شود.

page ستون و سطر (اعداد) (اعداد)

title عبارت

توضیحات

حدود ۴۴۱ و ۴۴۱

فصل ۵ و ۶

```
name_s segment para [combine] ['class']     ← نام استک
    DW 32 DUP(?)     ← چون مقدار اولیه مشخص نیست
name_s End
name_D segment para     ← دیتا سگمنت
    برچسب F DB مقدار اولیه ج اندازه
name_D End
name_c segment para     ← نام کد سگمنت
main
    SUB AX, AX
    MOV AX, name_D
    MOV DS, AX
        proc
            far → exe قالب برنامه
            near → com قالب برنامه
    Assume DS: name_D, SS: name_S, ....
        کد برنامه طبق الگوریتم
End proc
name_c Ends
```

بعد از این قرار دادن آدرس شروع دیتا سگمنت به درون ثبات DS می باشد

درون هر کد سگمنت می توان چندین برنامه یا proc بنویسید

Raz

Date: Subject: ۱۵

سوال

اندازه ss (استک سگمنت) چگونه مشخص می شود؟

با توجه به حجم ورودی و خروجی های برنامه نوشته شده و همچنین فراخوانی ها و انتقال پارامتر به درون زیر برنامه ها می توانیم حجم این سگمنت را مشخص نماییم اما نکته حائز اهمیت این است که اندازه یک سگمنت ۶۴ تا باشد که با توجه به حرف W در عبارت DW و عدد ۳۲ ظرفیت ۶۴ بدست می آید W=۲b

سوال

در قسمت DS (دیتا سگمنت) داده ها، ثابت ها و متغیرها چگونه تعریف می شوند

جواب: با توجه به الگوی اسم DN مقدار اولیه

به جای n هر کدام از حروف داخل پرانتز را می توان نوشت (B - W - F - D - T)

اگر بخواهیم داده های ثابت تعریف نمائیم از همین الگو استفاده می شود به جز این نکته که باید از کلمات زیر نیز بهره بگیریم

در مبنای ۱۰	D یا ۳۷
در مبنای ۱۶	H یا ۳۷
در مبنای ۸	O یا ۳۷
در مبنای باینری	B یا ۱۰۱۰
کاراکتر	'3'
رشته	" "

مفاهیم far و near چیست

کلمه far به معنی دور که بیان می کند proc تعریف شده می تواند از سگمنت های دیگر باشد اما در حالت near حتماً proc تعریف شده می باید در همان سگمنت تعریف شده، فراخوانی نیز گردد.

سوال

در یک کد سگمنت چه تعداد زیر برنامه proc می توانیم داشته باشیم

جواب: همانطوری که در زبان های سطح بالا وجود دارد برای وجود متدها در یک کلاس مفهوم شده است در زبان اسمبلی در یک کد سگمنت یک proc از نوع far وجود دارد و الباقی proc ها از نوع near می باشد.

Raz

Date: Subject: 11

سوال

مفهوم دستور Assume چیست؟

جواب: یک شبه دستوری است که برای آن هیچگونه کدی (کد ماشین) تولید نمی‌شود و تنها به اسمبلر یک راهنمایی می‌کند تا به شناخت ارتباط بین سگمنت‌ها با ثبات سگمنت

Assume DS:name_D , SS:neme_S , ES:nothing
دیتا سگمنت — استک سگمنت — هیچ چیز

سوال

اگر بخواهیم از محیط برنامه اسمبلر مستقیماً به درون سیستم عامل برویم چه دستوراتی باید نوشته شود.

Mov AH,4coH
int 21H

⟵ در انتهای PROC و قبل از End P نوشته می‌شود

تمرین

با توجه به ساختار برنامه نویسی EXE برنامه‌ای بنویسید که ۲ کاراکتر را از ورودی دریافت و پس آن دو را در کنار هم قرار دهد.

Mov AH,09H
Effective Load — Lea Dx, خروجی (آدرس متغیر را به ثبات می‌برد)
int 21H

تمرین

۱- برنامه بنویسید که یک عدد را از ورودی دریافت کند و سپس اعلام کند زوج است یا فرد

۲- برنامه‌ای بنویسید که یک عدد را بخواند معکوس آن را نشان دهد.

صفحه ۷۴ تا ۷۶ را کپی بگیر بیار

ورودی رشته

دستور ورودی:
Mov AH,0AH
LEA DX, اسم یا آدرس شروع
int 21H

دستور خروجی:
Mov AH,09H
LEA DX, داده (اسم متغیر رشته‌ای Massage)
int 21H

ورودی کاراکتر

دستور ورودی:
Mov AH,01H
int 21H

دستور خروجی:
Mov AH,02H
Mov DL, مقدار خروجی
int 21H

خروجی کاراکتر

Date: Subject: ۱۲

معرفی ثابت رشته‌ای

با استفاده از ظرفیت DB (Difine Byte) داده‌های رشته‌ای را معرفی می‌نماییم و برای مشخص شدن انتهای ثابت رشته‌ای از علامت $ بهره می‌گیریم.

اسم ثابت رشته‌ای DB 'کاراکترهای ثابت رشته‌ای', '$'

```
message DB 'Enter press any keys', '$'
```

متغیر رشته‌ای

متغیر رشته‌ای آرایه:
- اسم متغیر رشته‌ای LABEL Byte
- MaxLenDB عدد ⟵ حداکثر تعداد کاراکترهای یک رشته
- Act_Len DB عدد ⟵ تعداد کاراکترهای وارد شده
- اسم دستیابی به داده‌های متغیر DB عدد DUP (؟) ← مقدار اولیه یا صفر یا

(اگر به جای علامت سوال عدد $\neq$ (صفر) قرار دهیم تمام خانه‌ها با عدد صفر مقداردهی می‌شوند)

مثال

یک متغیر رشته‌ای برای دریافت فامیلی دانشجویان معرفی نمایید.

```
A Family LABEL Byte
  MaxLen DB 20
  Act_Len DB ?        ← برابر
  Fam     DB 20 DUP(' ')
```

تمرین

برنامه‌ای بنویسید که اسم دانشجو را دریافت و آنرا برعکس نشان دهد.

```
Mov SI,Act_Len ①
Mov CX,Act_Len ②
A20:
  Mov DL,name[SI]
  Mov AH,02H
  int 21H
Raz Dec SI
```