گزارش قسمت کدی دستورکار دهم ریزپردازنده و زبان اسمبلی

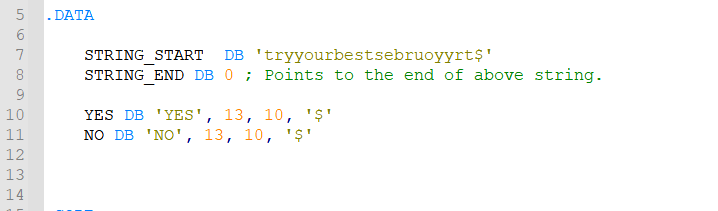
امیرحسین ادواری ۹۸۲۴۳۰۰۴- زهرا حیدری ۹۸۲۴۳۰۲۰

سوال اول)

قسمت اول:‌

ابتدا رشته مورد پردازش را تعریف و یک بایت نیز در انتهای آن تعریف میکنیم. پوینتر STRING\_START واضحا به آغاز رشته اشاره می­کند. پوینتر STRING\_END نیز به اولین بایت بعد از آخرین بایت رشته ورودی اشاره می­کند.

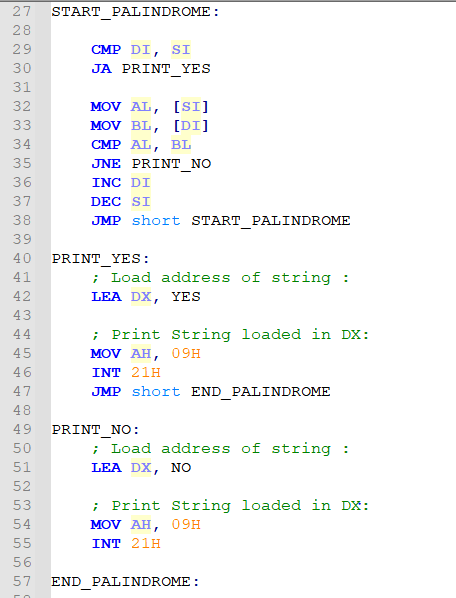
دو رشته برای چاپ عبارت YES و NO نیز تعریف می­کنیم.



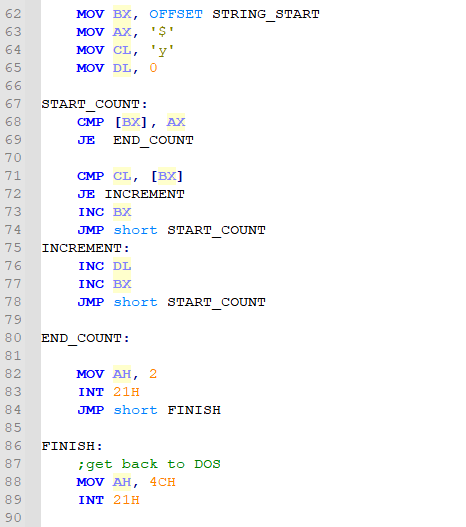
در قسمت کد،‌ پس از مقدار دهی پوینتر دیتاسگمنت، آدرس آغاز رشته را به DI و آدرس پایان رشته را (با کسر ۲ از STRING\_END آدرس پایان رشته به­دست می­آید)‌ به SI ‌ می­دهیم.

در یک حلقه متوالیا کاراکتری را که DI‌به آن اشاره می­کند با کاراکتری که SI به آن اشاره می­کند مقایسه می­کنیم. در صورت تساوی پیش می­رویم در غیر اینصورت NO‌ چاپ می­شود.

اگر دو پوینتر همدیگر را رد کنند (SI>DI) و همه مقایسه­ها منجر به تساوی شوند. YES چاپ می­شود.

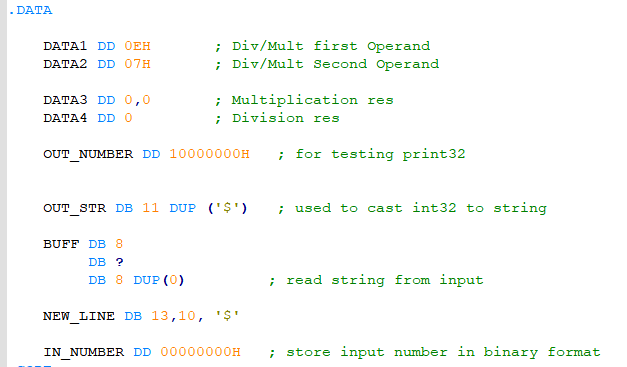


قسمت دوم:‌



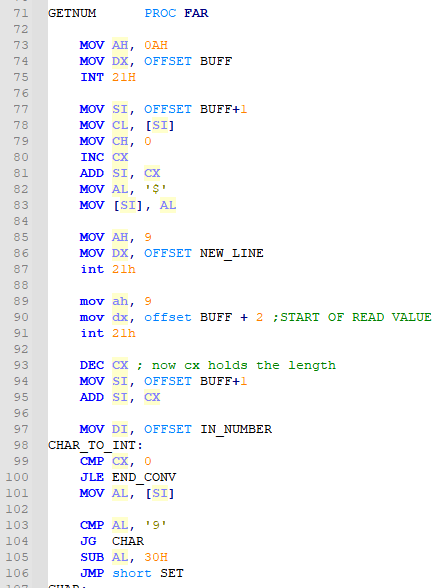
برای شمارش تعداد y‌ از ابتدای استرینگ (توسط BX که آدرس ابتدای رشته را نگه می­دارد)‌ پیمایش می­کنیم. اگر مقدار کاراکتری که BX به آن اشاره میکند معادل $ بود،‌ به انتهای رشته رسیده­ایم و شمارش تمام است. در غیر اینصورت اگر کاراکتر y بود مقدار DL و نیز BX را increment می­کنیم. و اگر کاراکتر خوانده شده y و $ نبود، صرفا BX را جلوتر می­بریم.

سوال دوم)



در ابتدا حافظه مورد نیاز برای عملکردهایی که در ادامه شرح داده می­شود را تعریف می­کنیم.

GETNUM:

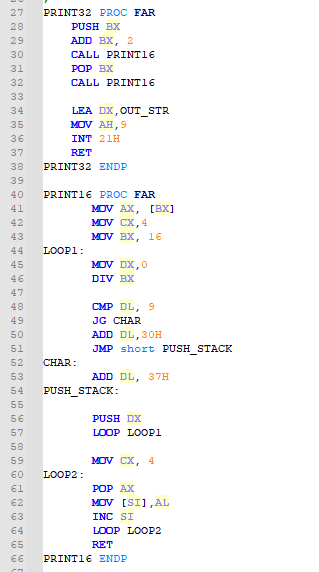


برای دریافت عدد از ورودی ابتدا آنرا به صورت رشته دریافت می­کنیم. سپس در انتهای آن $ قرار می­دهیم. (لازم به­ذکر است طبق هایلایتهای این اینتراپت، بعد از خواندن استرینگ طول آن را نیز در ابتدای بافر قرار میدهد. این مقدار را در CX قرار داده و از آن استفاده می­کنیم.

سپس SI را در انتهای رشته خوانده شده قرار می­دهیم و آنرا به صورت معکوس پیمایش می­کنیم.

در هر مرحله، ۲ کاراکتر از رشته را خوانده، هرکدام را به معادل عددی شان (که بین ۰ تا F می­باشد. روال تبدیل بدینگونه­است که اگر کاراکتر بیشتر از مقدار ‘9’ بود ۳۷ را از آن کم می­کنیم درغیر اینصورت ۳۰ را از آن کم می­کنیم) یکی از آنها را در نیبل پایینی و دیگری را در نیبل بالایی قرار می­دهیم. سپس این مقدار ۸ بیتی را که در هر نیبل مقدار عددی یک کاراکتر قرارگرفته­است را در آدرسی که DI‌ به آن اشاره می­کند قرار می­دهیم. در نهایت DI را جلوتر برده و SI را عقبتر می­بریم. (SI هر زمان که کاراکتر خوانده می­شود بایستی یک بایت عقبتر بیاید)‌ بدین­ترتیب عدد ورودی کاراکتری را به صورت هگز (که قابلیت شرکت در محاسبات را دارد)‌ ذخیره کردیم.

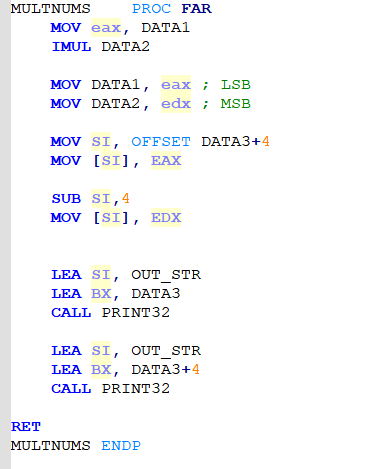
PRINTNUM:



یک سابروتین PRINT16 داریم که در این سابروتین، آدرس یک عدد ۱۶ بیتی را گرفته هر مرتبه در یک حلقه با ۴ تکرار، در هر مرحله عدد را تقسیم بر ۱۶ کرده، باقیمانده را که مقداری عددی­است به مقدار اسکی (کاراکتری) آن تبدیل می­کنیم ( روال کار در تبدیل بدین­گونه است که اگر مقدار آن بزرگتر از ۹ بود با ۳۷ و در غیر اینصورت با ۳۰ جمع می­کنیم)‌ و سپس در استک قرار می­دهیم. درنهایت ۴ مقدار وارد استک شده را یکی پس از دیگری از آن خارج می­کنیم و در یک استرینگ قرار می­دهیم ( توسط یک پوینتر SI که پس از نوشتن هر کاراکتر یک واحد جلو برده میشود (اضافه می­شود))

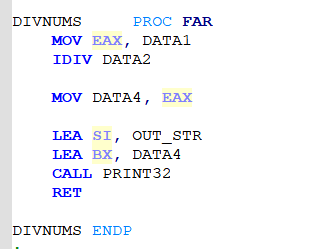
در روتین PRINT32 دو مرتبه روتین PRINT16 را کال می­کنیم. یکبار برای ۱۶ بیت بالایی عدد ۳۲ بیتی مدنظر و بار دیگر برای ۱۶ بیت پایین آن. بدین­ترتیب عددی که در حافظه ذخیره کرده­بودیم را به صورت یک رشته در فرمت هگز چاپ می­کنیم.

MULNUMS:



برای انجام عملیات ضرب DATA1 را در EAX‌ قرار می­دهیم. سپس عملیات IMUL را با عملوند DATA2 انجام می­دهیم. ۳۲ بیت پایین جواب در EAX و ۳۲ بیت بالایی آن در EDX قرار می­گیرد. با این حساب هرکدام از این ۳۲ بیت را در نیمی­از متغیر ۶۴ بیتی DATA3 قرار می­دهیم. در نهایت آن­هارا با روتین PRINT32 چاپ می­کنیم.

DIVNUMS:



برای عملیات تقسیم، DATA1 را در EAX قرار داده (مقسوم) سپس IDIV را با عملوند DATA2 (مقسوم علیه)‌ اجرا می­کنیم. در نهایت خارج­قسمت در EAX قرار میگیرد، آنرا در DATA4‌ قرار می­دهیم. در آخر نیز این دیتای ۳۲ بیتی را پرینت می­کنیم.