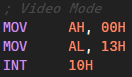
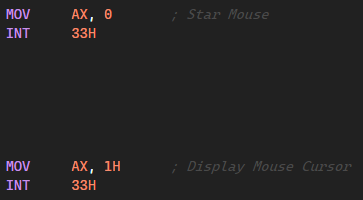
**امیرحسین احمدی 97522292 – تمرین پنجم**

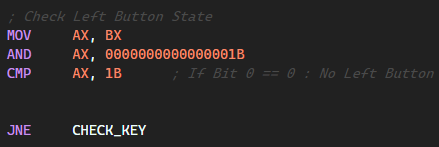
1. برای این سوال ابتدا نیاز است که برای استفاده از رنگ ها به ویدیو مود برویم.

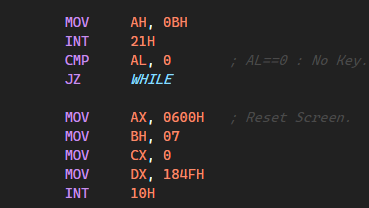


سپس برای استفاده از موس نیاز است تنظیمات زیر را انجام دهیم.

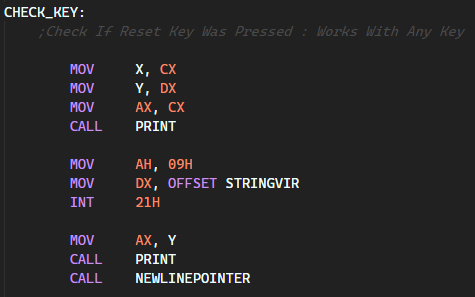


سپس در یک while هر دفعه چک میکنیم که آیا از صفحه کلید دکمه ای (هر دکمه ای) زده شده است، در این صورت الگوریتم پایان میابد و صفحه نمایش را پاک میکنیم.

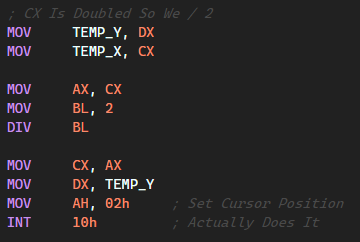




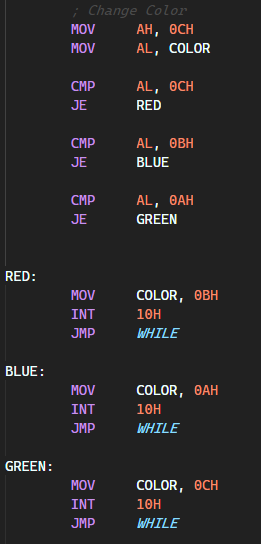
همچنین همیشه در ابتدای CHECK\_KEY مختصات خانه ای که در حال حاضر در آن هستیم را چاپ میکنیم.



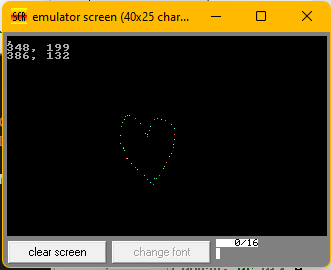
در ادامه در هر مرحله محله موس را دریافت کرده و نگه داشته تا در ادامه آن را رنگ کنیم.



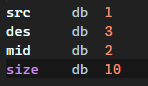
برای رنگ کردن نیز ابتدا کد رنگ قرمز را در color قرار داده و در هر مرحله با رنگ کردن یک نقطه، رنگ ها را تغییر میدهیم.



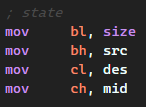
خروجی کد را نیز میتوان در زیر دید.



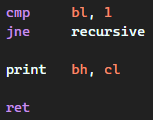
1. برای سوال دو ابتدا متغییر های src, des, mid و size را تعریم میکنیم که به ترتیب برابر میله ی مبدا، میله ی مقصر، میله ی وسط و تعداد دیسک ها هستند.



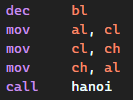
سپس متغیر های بالا را درون bl, bh, cl, ch میریزیم که قرار است در هر مرحله از الگریتم استیت مارا نگه دارند.



حال تابع بازگشتی hanoi را صدا میزنیم. در این تابع ابتدا چک میکنیم که اگر تعداد دیسک ها برابر 1 بود، همان دیسک را از میله ی مبدا به مقصد چاپ کند و الگوریتم پایان بپزیرد.



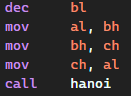
در غیر این صورت به این صورت باید عمل کنیم. ابتدا باید 9 دیسک اول را به میله ی وسط برده و سپس دیسک آخر را به میله ی مقصد ببریم و در ادامه 9 دیسک را به میله ی مقصد ببریم. حال برای پیاده سازی ابتدا باید تعداد میله ها را یکی کم کنیم، همچنین جای میله ی وسط و مقصد را عوض کنیم تا بتوان با صدا زدن دوباره تابع hanoi، 9 دیسک اول را به میله وسط برد.



سپس میله آخرین دیسک میله مبدا را به میله مقصد میبریم.



و بعد دوباره تعداد دیسک ها را 9 در نظر گرفته و این بار جای میله مبدا را با میله وسط عوض میکنیم تا بتوانیم 9 دیسک را از وسط به مقصد ببریم و الگوریتم ما تمام است.

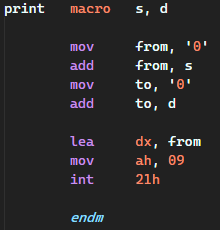


به این نکته توجه داشته باشید که از آنجایی که ممکن است در صدا زدن توابع متغییر ها تغییر کنند، هر بار قبل از صدا کردن تابعی متغییر هایی که استیت مارا نگه میداشتند را درون استک پوش کرده و بعد از انجام آن پاپ میکنیم.

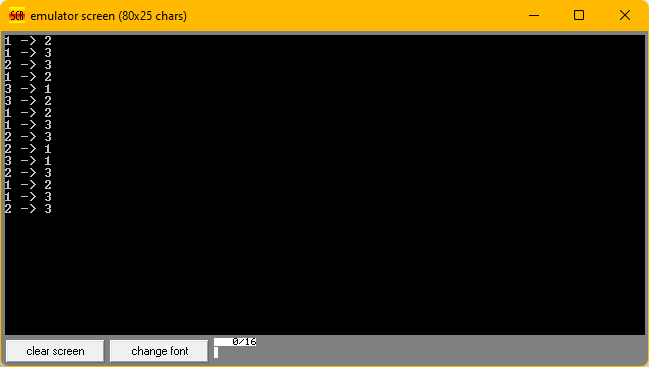




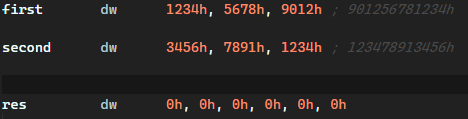
همچنین تابع print با استفاده از ماکرو پیاده سازی شده است به این صورت که میله ی مبدا و مقصد را برای جا به جایی یک دیس میگیرد و جا به جایی آن را با استفاده از یک استرینگ چاپ میکند.



خروجی سوال دو برای 10 دیسک بسیار طولانی است، در زیر خروجی را برای 4 دیسک مشاهده میکنید.

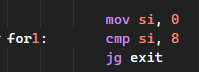


1. برای سوال 3، دو عدد 48 بیتی خود را به صورت یک لیست سه تایی از اعداد 16 بیتی در نظر میگیریم که عدد اول نشان دهنده کم ارزش ترین 16 بیت است. برای ریزالت نیز یک لیست 6 تایی مانند دو عدد ورودی درنظر میگیریم.



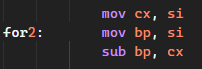
حال به این صورت عمل میکنیم که متغییر res را با شروع از اولی 16 بیتی اش به ترتیب پر میکنیم تا نتیجه بدست بیاید. برای این کار متغییر si را در نظر میگیریم. فقط توجه کنید از آن جایی که لیست ها از اعداد 16 بیتی هستند برای حرکت روی لیست باید si را دوتا دوتا افزایش دهیم که بتواند به عدد بعدی برود.

با توجه به توضیحات بالا از si مساوی صفر یک فور زده و هر وقت که si بزرگتر از 8 شد الگوریتم تمام است.

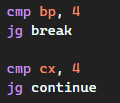


برای انجام عمل ضرب به این صورت باید عمل کنیم که در هر مرحله res[i] باید برابر شود با مجموع first[j] و second[k] هایی که j به علاوه k برابر با i باشند و همچنین در صورت وجود کری باید آن را به خانه i + 1 ببریم.

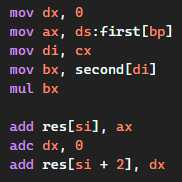
برای این کار متغییر cx را که قرار است به خانه های عدد دوم اشاره کند را ابتدا برابر si قرار میدهیم و در هر مرحله bp را که قرار است روی عدد اول حرکت کند برابر با si منهای cx قرار میدهیم. همچنین در هر مرحله قرار است cx را یکی یکی کم کرده (و به طبع bp زیاد میشود) تا بتوانیم تمام j و k های گفته شده در بالا را بدست آوریم.



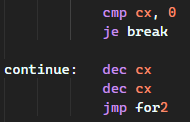
از آن جایی که cx میتواند فقط از 2 تا 0 مقدار بپزیرد، در صورت بالاتر بودن آن باید cx را کم کرده و الگوریتم را دوباره اجرا کنیم. همچنین bp نیز اگر مقداری بیشتر از 2 پیدا کند، الگوریتم تمام است و باید به سراغ si بعدی برویم.



حال هر بار first[bp] و second[cx] را درنظر گرفته و حاصل ضرب آن ها را با res[si] جمع میکنیم و باقی آن را با res[si + 1] جمع میکنیم.



در آخر نیز چک میکنیم که در صورت صفر بودن cx به پایان الگوریتم رسیده ایم و باید به si بعدی برویم و در غیر این صورت با زیاد کردن cx به حالات دیگر میرویم.



خروجی الگوریتم را میتوانید در زیر ببینید.

