**به نام خدا**

**پروژه درس ساختار و زبان کامپیوتر**

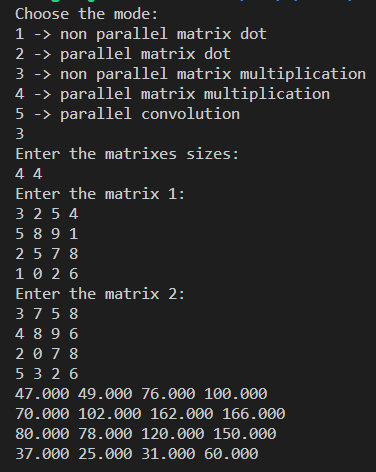
**استاد جهانگیر**

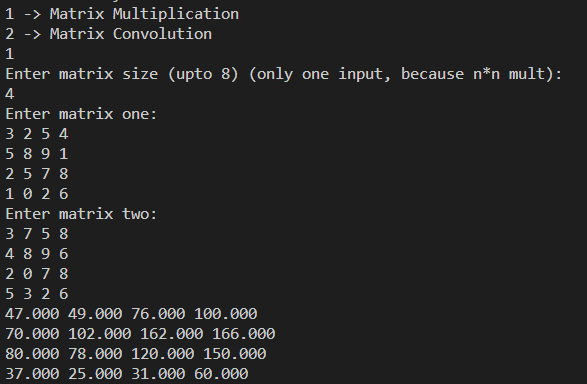
**شرح پروژه:**

1. **برنامه اسمبلی ضرب دو ماتریس n\*n اعداد ممیز شناور 32 بیتی به روش معمولی، یعنی ضرب  و جمع متوالی درایه‌های متناظر دو ماتریس و مقایسه سرعت اجرای این برنامه با استفاده از دستورات برداری (موازی) پردازنده  را برای  پردازنده 8086 در دو حالت n=3 و n=5 بنویسید و با هم مقایسه کنید. همین طور با زمان اجرای برنامه‌ی ضرب ماتریس به زبان سطح بالا مقایسه و نقد کنید.**

برای این بخش، برنامه خواسته شده پیاده سازی شده است، و با choose mode سه میتوان به صورت غیر موازی، و چهار به صورت موازی دو ماتریس n\*n را در هم ضرب کرد.

برای بدست آوردن زمان اجرای قابل قیاس، فرآیند ضرب کردن را در یک حلقه، 1 میلیون بار تکرار کرده ایم.

نتایج بدست آمده:



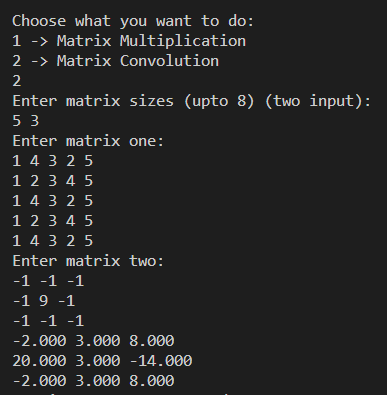
خروجی با زبان سطح بالا (پایتون)

زمان اجرا: 4.531 ثانیه

خروجی با زبان اسمبلی

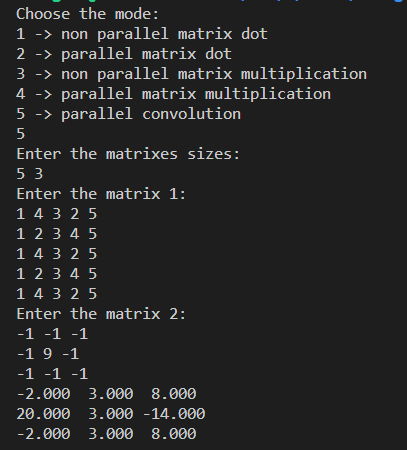
زمان اجرا: 0.067 ثانیه (غیر موازی) – 0.044 ثانیه (موازی)

1. **سپس برنامه خود را برای محاسبه یک تابع Convolution-2D دلخواه به کار ببرید و زمان اجرای برنامه کامل را با برنامه‌(های) مشابه موجود یا خودتان مقایسه کنید. توضیح اینکه عمل Convolution (به زبان ساده) یک تابع را بر روی محور افقی از روی یک تابع دیگر عبور می‌دهد و در هر نقطه برخورد (یا در نقاط گسسته) حاصل ضرب این دو تابع را محاسبه  و ثبت می‌کند. میتوانید یک تابع را با ماتریس 5 در 5  ودیگری را 3 در 3 در نظر بگیرید.**

برای این بخش هم با choose mode پنج می­توان در همان فایل، میتوان عمل کانولوشن را روی دو ماتریس ورودی اعمال کرد. این تابع (کانولوشن) با استفاده از dot product ای که به صورت موازی پیاده سازی شده است (choose mode دو)، زده شده است (توضیحات تکمیلی در کد موجود می­باشد).  
در این بخش سایز ماتریس فیلتر (matrix2) نمی­تواند از ماتریس اصلی (اول) بیشتر باشد. سایز ماتریس ها هم نهایتا 8 می­تواند باشد.  
برای بدست آوردن زمان اجرای قابل قیاس، فرآیند ضرب کردن را در یک حلقه، 100 هزار بار تکرار کرده ایم.  
نتایج بدست آمده:

خروجی با زبان سطح بالا (پایتون)

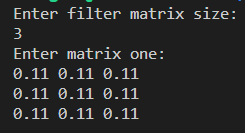
زمان اجرا: 1.692 ثانیه



خروجی با زبان اسمبلی

زمان اجرا: 0.192 ثانیه

1. **برنامه خود را برای محاسبه یک تابع Convolution-2D دلخواه (مثلاً برای یک کار پردازش تصویر یا هوش مصنوعی)به کار ببرید.**

در این بخش، یک کد پایتون نوشته شده که ابتدا عکس را به یک ماتریس تبدیل میکند (gray scaled). سپس ابتدا ماتریس فیلتر (کرنل) را ورودی میگیرد. سپس در هر مرحله یک ماتریس 8\*8 را از ماتریس درآمده از عکس، در input.txt میریزد و سپس کد اسمبلی را با این ورودی ها ران میکند. بعد، خروجی را هربار در انتهای فایل result.txt می­نویسد. در نهایت این فایل تحلیل می­شود و عکس خروجی را می­سازد (ذخیره میکند).  
نمونه با فیلتر blur:

شکل 2 تصویر مات شده (خروجی)

شکل 3 تصویر ورودی

شکل 1 ورودی کد (ماتریس فیلتر)

**توضیحات فایل ها:**

* **Project.asm:** فایل اصلی پروژه اسمبلی
* **Asm\_io.asm:** مجموعه توابع برای ورودی-خروجی
* **Driver.c:** فایل سی برای کال کردن تابع main پروژه
* **High\_level:** کد پایتون برای محاسبه ضرب و کانولوشن دو ماتریس (برای محاسبه تفاوت زمانی با اسمبلی)
* **Image\_proccessor:** کد پایتون برای استفاده از کد اسمبلی برای اعمال فیلتر بر روی تصاویر
* **Input\_db:** بانک تست کیس ها
* **project\_no\_extra\_print:** همان فایل پروژه است، صرفا حلقه های محاسبه زمان، و پرینت دیالوگ ها (مانند enter matrix 1:) را ندارد و صرفا ورودی می­گیرد و خروجی را پرینت میکند. از این فایل در Image\_proccessor استفاده شده است.
* **Run.sh:** فایل ران کردن کد های اسمبلی (اسکریپت برای کامپایل، اسمبل و لینک کردن فایل های لازم). اسم فایلی که میخواهد ران بشود را ورودی میگیرد (برای مثال: ./run.sh project). با این فایل میتوان کد ها را فقط اسمبل و لینک کرد یا ران کرد و یا زمان اجرا را هم گرفت.
* **Result.jpg:** خروجی Image\_proccessor
* **Xmpl.jpg:** ورودی Image\_proccessor