صادق سرگران 401106039

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:

```
Choose the mode:
1 -> non parallel matrix dot
2 -> parallel matrix dot
3 -> non parallel matrix multiplication
4 -> parallel matrix multiplication
5 -> parallel convolution
3
Enter the matrixes sizes:
4 4
Enter the matrix 1:
3 2 5 4
5 8 9 1
2 5 7 8
1 0 2 6
Enter the matrix 2:
3 7 5 8
4 8 9 6
2 0 7 8
5 3 2 6
47.000 49.000 76.000 100.000
70.000 102.000 162.000 166.000
80.000 78.000 120.000 150.000
37.000 25.000 31.000 60.000
```

خروجی با زبان اسمبلی

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

```
1 -> Matrix Multiplication
2 -> Matrix Convolution
1
Enter matrix size (upto 8) (only one input, because n*n mult):
4
Enter matrix one:
3 2 5 4
5 8 9 1
2 5 7 8
1 0 2 6
Enter matrix two:
3 7 5 8
4 8 9 6
2 0 7 8
5 3 2 6
47.000 49.000 76.000 100.000
70.000 102.000 162.000 166.000
80.000 78.000 120.000 150.000
37.000 25.000 31.000 60.000
```

خروجی با زبان سطح بالا (پایتون)

زمان اجرا: 4.531 ثانیه

2- 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:

```
Choose the mode:
1 -> non parallel matrix dot
2 -> parallel matrix dot
3 -> non parallel matrix multiplication
4 -> parallel matrix multiplication
5 -> parallel convolution
5
Enter the matrixes sizes:
5 3
Enter the matrix 1:
1 4 3 2 5
1 2 3 4 5
1 4 3 2 5
1 2 3 4 5
1 4 3 2 5
Enter the matrix 2:
-1 -1 -1
-1 9 -1
-1 -1 -1
-2.000  3.000  8.000
20.000  3.000 -14.000
-2.000  3.000  8.000
```

خروجی با زبان اسمبلی

زمان اجرا: 0.192 ثانیه

```
Choose what you want to do:
1 -> Matrix Multiplication
2 -> Matrix Convolution
2
Enter matrix sizes (upto 8) (two input):
5 3
Enter matrix one:
1 4 3 2 5
1 2 3 4 5
1 4 3 2 5
1 2 3 4 5
1 4 3 2 5
Enter matrix two:
-1 -1 -1
-1 9 -1
-1 -1 -1
-2.000 3.000 8.000
20.000 3.000 -14.000
-2.000 3.000 8.000
```

خروجی با زبان سطح بالا (پایتون)

زمان اجرا: 1.692 ثانیه

3- 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:

```
Enter filter matrix size:
3
Enter matrix one:
0.11 0.11 0.11
0.11 0.11 0.11
0.11 0.11 0.11
```

شکل 3 ورودی کد (ماتریس فیلتر)

شکل 2 تصویر ورودی

شکل 1 تصویر مات شده (خروجی)

صادق سرگران 401106039

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