به نام خدا

پروژه درس ساختار و زبان کامپیوتر

ترم1-1404

سعید نوفرستی 403106838

-هدف این پروژه پیاده‌سازی و مقایسه‌ی **کانولوشن دوبعدی تصویر (2D Convolution)** به دو روش است:

۱. پیاده‌سازی ساده با C (Baseline)
۲. پیاده‌سازی بسیار سریع با Assembly + AVX SIMD

سپس:
- مقایسه زمان اجرا (Benchmark)
- محاسبه Speedup
- Zero Padding
- تشخیص لبه با Sobel
- تشخیص شی (Object Recognition)
- اجرای خودکار روی صدها تصویر

این برنامه روی تصاویر pgm کار میکند پس ابتدا نیاز داریم این دستور را در ترمینال لینوکس وارد کنیم

```
sudo apt update
sudo apt install imagemagick
```

سپس هر تصویر دلخواه را با دستور

```
convert input.jpg -colorspace Gray input.pgm
```

به تصویر pgm تبدیل میکنیم

ساختار کلی برنامه به اینصورت است

```
├── main.c                  # فایل اصلی برنامه
├── conv.c                  # نسخه C کانولوشن
├── conv.asm                # نسخه Assembly + AVX
├── conv.h                  #هدر
├── timer.c                 # اندازه‌گیری زمان
├── object_recognition.c    # پردازش 100/400 تصویر
├── inputs/                 # تصاویر ورودی
└── Makefile                # پیکر بندی اجرایی
```

فایل c برای محاسبه تایم(time.c)

timer.c

```c
#include <time.h>

double now(){
    struct timespec ts;
    clock_gettime(CLOCK_MONOTONIC, &ts);  // گرفتن تایم از MONOTONIC
    return ts.tv_sec + ts.tv_nsec*1e-9; // برگرداندن تایم به واحد ثانیه
}
```

فایل پیکر بندی برای اجرای راحت تر برنامه (Makefile)

Makefile

```
# کامپایلر C
CC=gcc

# اسمبلر NASM
ASM=nasm

# فلگ‌های کامپایل:
# -O3 : بیشترین سطح بهینه‌سازی
# -mavx : فعال‌سازی دستورات AVX
CFLAGS=-O3 -mavx


# تارگت پیش‌فرض -> ساخت فایل اجرایی main
all: main


# تبدیل فایل اسمبلی به آبجکت
conv.o: conv.asm
    $(ASM) -f elf64 conv.asm -o conv.o


# کامپایل و لینک کل پروژه
# -lm برای توابع ریاضی مثل fabs,...
main: main.c conv.c  object_recognition.c timer.c conv.o
    $(CC) $(CFLAGS) main.c conv.c object_recognition.c timer.c conv.o -lm -o main



# پاک کردن فایل‌های ساخته شده
clean:
    rm -f main *.o output_c.pgm output_asm.pgm
```

کرنل های متناسب با فیلتر های مختلف

```
// Edge_Detection
float ker_Edge[9] = {
    -1,-1,-1,
    -1, 8,-1,
    -1,-1,-1
};

// Blur
float ker_Blur[9] = {
    1.00f/9.00f,1.00f/9.00f,1.00f/9.00f,
    1.00f/9.00f,1.00f/9.00f,1.00f/9.00f,
    1.00f/9.00f,1.00f/9.00f,1.00f/9.00f
};

// Sharpen
float ker_sharpen[9] = {
0, -1,  0,
-1,  5, -1,
0, -1,  0
};
```

فایل conv.h برای مشخص کردن پروتوتایپ تابع conv2d به زبان c و asm

```
13
14  void conv2d_c(float* in, float* out, float* kernel,
15                int w, int h, int k);
16
17
18  void conv2d_asm(float* in, float* out, float* kernel,
19                  int w, int h, int k);
20
```

کد توابع در فایل های conv2d.c و conv2d.asm موجود است
حسن asm نسبت به c استفاده از AVX و توابع برداری است
به عنوان مثال در تصویر زیر نمونه ان را میبینیم :

```
; Row 0
; سطر موقعیت کنونی
vmovups ymm1, [r12-4]
vmulps  ymm1, ymm1, ymm7 ; k5 * وزن پیکسل
vaddps  ymm0, ymm0, ymm1 ; با جمع نتیجه sum

vmovups ymm1, [r12]
vmulps  ymm1, ymm1, ymm8 ; k4 * وزن پیکسل
vaddps  ymm0, ymm0, ymm1 ; با جمع نتیجه sum

vmovups ymm1, [r12+4]
vmulps  ymm1, ymm1, ymm9 ; k3 * وزن پیکسل
vaddps  ymm0, ymm0, ymm1 ; با جمع نتیجه sum
```

فایل object_recognition.h برای مشخص کردن پروتوتایپ
تابع object_recognition.c که برای پردازش چند صد تصویر
به زبان asm, c و تشخیص شی استفاده میشود, است

object_recognition.h

```
1  #pragma once // جلوگیری از چندبار include (جلوگیری از خطای تکرار تعریف) شدن فایل هدر
2  // اجرای تشخیص شی روی چند تصویر
3  // folder : پوشه تصاویر (مثلاً inputs)
4  // count  : تعداد تصاویر
5  void objectRecognition(const char* folder, int count);
6
```

حالا به سراغ اجرای برنامه روی تصویر دلخواه میرویم
این برنامه چهار مد کاری دارد

۱. blur (تک تصویره)

۲. sharpen (تک تصویره)

۳.Edge_Detection (تک تصویره)

۴.object recognition (روی ۴۰۰ تصویر دارای شی یا بدون شی)

برای شروع یک تصویر(pgm) را انتخاب میکنم برای مثال تصویر زیر:

1. اجرای فیلتر blur

```
make clean
make
./main Blur input30.pgm
```

خروجی ASM و C :

که به وضوح نسبت به تصویر اول blur شده است

خروجی کد در ترمینال به این شکل بود:

```
saeed@saeed-Vivobook-ASUSLaptop-K3605VU-K3605VU:~/me/Convolution$ ./main Blur input30.pgm
C time = 0.005142
ASM time = 0.000719
speedup : 7.153068
error : 0.000000
```

که نشان میدهد سرعت کار با اسمبلی 7.15 برابر کدC است و ارور صفر نشان دهنده این است که خروجی asmوc دقیقا مشابه هم هست با این تفاوت که سرعت asm بسیار بیشتر بوده است

2.اجرای فیلتر Sharpen

حالا تصویر blur شده در قسمت قبل با بار نام input.pgm ذخیره کرده و روی ان فیلتر sharpenرا اجرا میکنیم:

```
make clean
make
./main Sharpen input.pgm
```

اینبار خروجی در ترمینال به این شکل بود

```
saeed@saeed-Vivobook-ASUSLaptop-K3605VU-K3605VU:~/me/Convolution$ ./main Sharpen input.pgm
C time = 0.005701
ASM time = 0.000936
speedup : 6.088460
error : 0.000000
```

و خروجی تصویر به شکل زیر است که به وضوح از تصویر blurشده قبل شارپ تر شده است

C

ASM

3.اجرای فیلتر Edge_Detection

```
make clean
make
./main Edge_Detection input30.pgm
```

خروجی کد

```
saeed@saeed-Vivobook-ASUSLaptop-K3605VU-K3605VU:~/me/Convolution$ ./main Edge_Detection input30.pgm
C time = 0.004440
ASM time = 0.000866
speedup : 5.127548
error : 0.000000
```

تصویر خروجی

ASM

C

لبه های تصویر مشخص شده هست

-حالا سراغ کار نهایی میرویم که تشخیص شی در ۴۰۰ تصویر مختلف هست اینبار در پوشه inputs تصاویر مختلف که ۲۰۰ تصویر دارای شی و ۲۰۰ تصویر تصاویر سفید و خاکستری و مشکی و... بدون شی هستند را قرار میدهیم تا روی این تصاویر این ازمایش را انجام دهیم

make clean
make
./main object_recognition
خروجی کد به این صورت بود

```
saeed@saeed-Vivobook-ASUSLaptop-K3605VU-K3605VU:~/me/Convolution$ ./main object_recognition
REPORT
C   total time  : 2.702299 sec
ASM total time  : 0.765157 sec
Speedup         : 3.53x
Objects (C)     : 199 / 400
Objects (ASM)   : 199 / 400
```

حتی با وجود اینکه در تایم اسمبلی تایم zero paddingهم محاسبه شده باز هم حدود 3.5برابر سرعت بیشتری به ما میدهد که شگفت انگیز است
اما میبینیم که خروجی از لحاظ تشخیص تعداد تصویر برای هر دو کد c , asm یکسان است
دقت محاسبه به این صورت است که :
۱ تصویر که دارای شی بوده به اشتباه بدون شی تشخیص داده شده و ۳۹۹ تصویر درست تشخیص داده شده است:
پس با دقت 399/400*100 درصد محاسبه انجام شده

یعنی دقت برابر بوده با :

% 99.75

این پروژه در لینک زیر در گیت هاب هم دردسترس است

https://github.com/SSN4444/Convolution.git

با تشکر از استاد گرامی دکتر جهانگیر