CS 270

Homework 01

Due 09/14/09

Colby Blair

I. Goal/Motivation

II. Code

See attached files

III. Data

I decided not to include the entire ELF file here, but on a subsequent page, due to its size. I had 4 functions: arr_sum, find_max, find_min, and main. There function is pretty self explanatory in the context of arrays, and find_min and find_max are almost identical in c operations. It seems, from the data, that they are almost identical in assembly/machine code as well.

COMPILE IN NORMAL MODE (gcc -Wall -o <app.c> <app>)

FILE: main

Functions

Name	Size	Address
arr_sum	54	080483c4
find_max	70	080483fa
find_min	70	08048440
main	150	08048486

Program info

Entry Point -

Name: .text

Location: 0x8048310

Subroutines from stdio.h:

Name	Location
printf	reference to GLIBC_2.0, unknown

Data Sections:

Name	Size	Location
arr	40	0804a040
SIZE	4	080485e0

COMPILE IN O3 MODE (gcc -Wall -O3 -o <app.c> <app>)

FILE: mainO3

Functions

Names	Size	Address
arr_sum	64	080483f0
find_max	119	08048430
find_min	119	080484b0
main	757	08048530

Program info

Entry Point -

Name: .txt

Location: 0x8048330

Subroutines from stdio.h:

Name	Location
printf	reference to GLIBC_2.3.4, unknown

Data Sections:

Name	Size	Location
arr	40	0804a040
SIZE	4	0804891c

Directory list:

myriad360@colbys-desktop:~/CS_270/hw01$ ls -talh

total 124K

drwxr-xr-x 2 myriad360 myriad360 4.0K 2009-09-12 15:15 .

-rw-r--r-- 1 myriad360 myriad360 380 2009-09-12 15:15 main.c

-rw-r--r-- 1 myriad360 myriad360 13K 2009-09-12 15:09 elfdataO3.txt

-rwxr-xr-x 1 myriad360 myriad360 9.2K 2009-09-12 15:08 mainO3

-rw-r--r-- 1 myriad360 myriad360 13K 2009-09-12 15:06 #elfdata.txt#

-rw-r--r-- 1 myriad360 myriad360 13K 2009-09-12 14:46 elfdata.txt

-rwxr-xr-x 1 myriad360 myriad360 9.2K 2009-09-12 14:44 main

-rw-r--r-- 1 myriad360 myriad360 150 2009-09-12 14:43 find_min.h

-rw-r--r-- 1 myriad360 myriad360 150 2009-09-12 14:43 find_max.h

-rw-r--r-- 1 myriad360 myriad360 119 2009-09-12 14:42 arr_sum.h

-rw-r--r-- 1 myriad360 myriad360 5.1K 2009-09-12 14:12 elfdata02.txt

-rw-r--r-- 1 myriad360 myriad360 13K 2009-09-12 13:51 elfdata.txt~

-rw-r--r-- 1 myriad360 myriad360 725 2009-09-12 13:50 main.c~

drwxr-xr-x 3 myriad360 myriad360 4.0K 2009-09-12 13:14 ..

IV. Revision of Data/Conclusion

Overall, this program was pretty tiny. It was a simple function, and doing the optimization didn't do much for system memory. Each file was at 9.2 K, in fact. But this is actually a good thing in this case; the O3 optimization did expand my functions by a lot by unrolling the loops (arr_sum increased by a factor of 1.18, find_max and find_min by a factor of 1.7, and main by a factor of 5.04!). This was a negligible increase on the scale of the OS. If this program was used repetitively by some process, however, the increase in performance could be substantial with the relative level of change done to the program. the O3 option would take much more advantage of cache.

Take this into consideration:

```
myriad360@colbys-desktop:~/CS_270/hw01$ time ./main
...
real    0m0.005s
user    0m0.000s
sys    0m0.005s
myriad360@colbys-desktop:~/CS_270/hw01$ time ./main03
...
real    0m0.004s
user    0m0.000s
sys    0m0.004s
```

The increase is certainly negligible in a one time use. And if the program was larger, we could probably get a better definition of time increase, instead of being so close to the minimal sample rate here. However, we can theorize that if the times above are accurate, significant gains would be seen with the O3 compilation if it was used to find array data and compare it 24 hours a day, on a massive process job.

In conclusion, the ELF file format has lifted a vial of ambiguity from my knowledge between the compiler and assembly code. I have liked both, but it is the understanding of this area that truly regains my confidence in what is going on with system software. Knowing the processes of O3 (like unrolling finite loops) and the ELF format, it is clear what the advantages and disadvantages are. Low process time vs. low memory usage, taking advantage of cache vs. debugging ability, etc. How to build and compile programs really depends on the application and level of refinement a program has gone through in its life cycle.