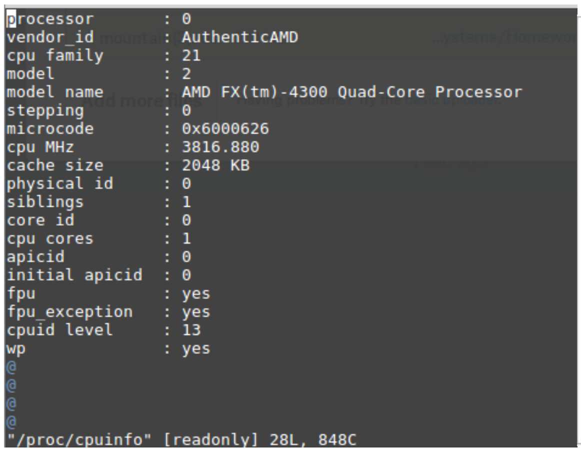
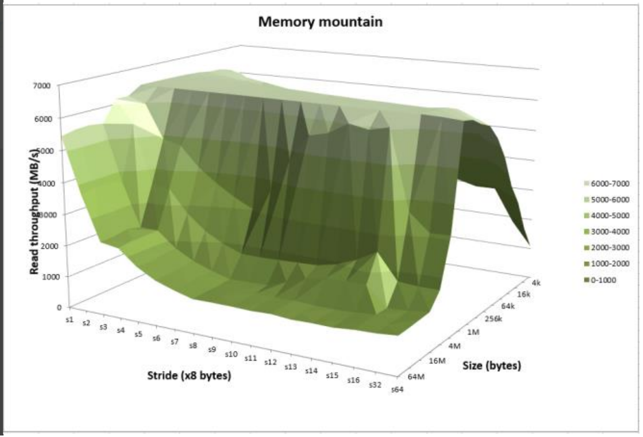
1. 424-8

Linux VM specs

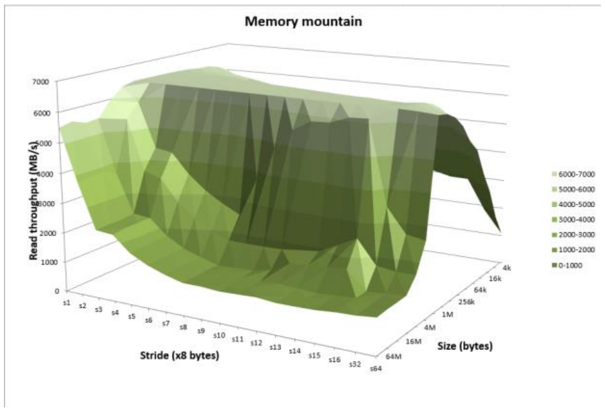


```
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 21
model           : 2
model name      : AMD FX(tm)-4300 Quad-Core Processor
stepping        : 0
microcode       : 0x6000626
cpu MHz         : 3816.880
cache size      : 2048 KB
physical id     : 0
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
@
@
@
@
"/proc/cpuinfo" [readonly] 28L, 848C
```

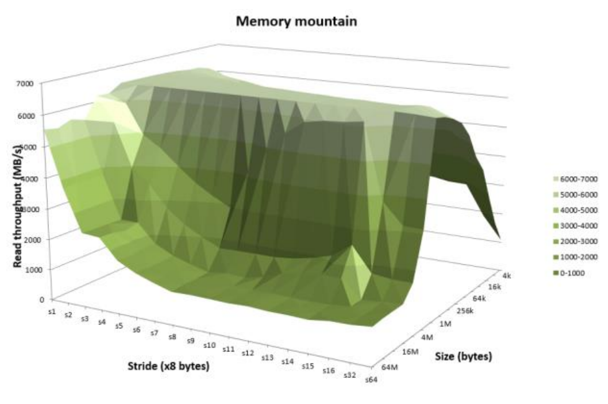
Compiled with -o3



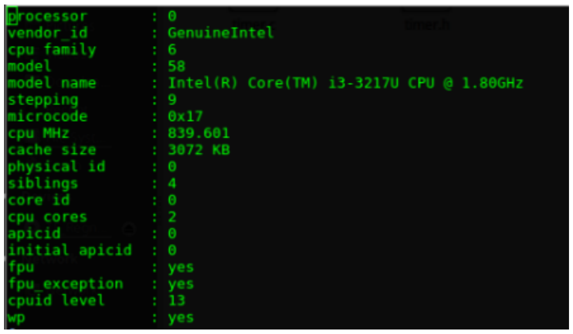
Compiled with -o2



Compiled with -o1

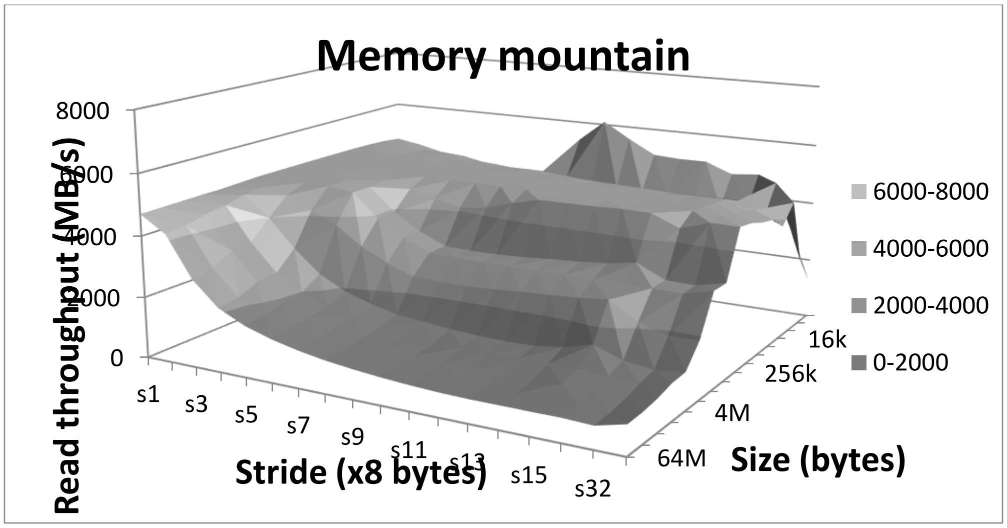


Linux Mint Laptop Specs

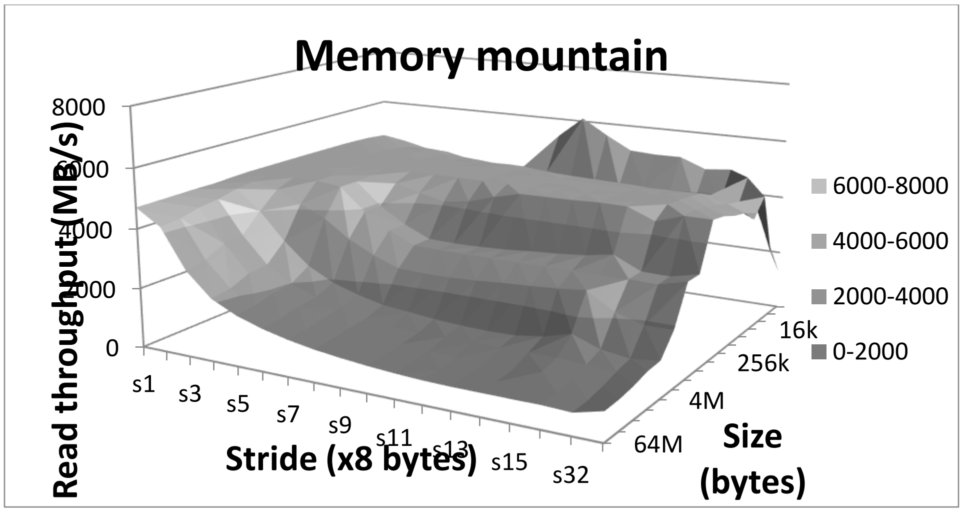


```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 58
model name      : Intel(R) Core(TM) i3-3217U CPU @ 1.80GHz
stepping        : 9
microcode       : 0x17
cpu MHz         : 839.601
cache size      : 3072 KB
physical id     : 0
siblings        : 4
core id         : 0
cpu cores       : 2
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
```

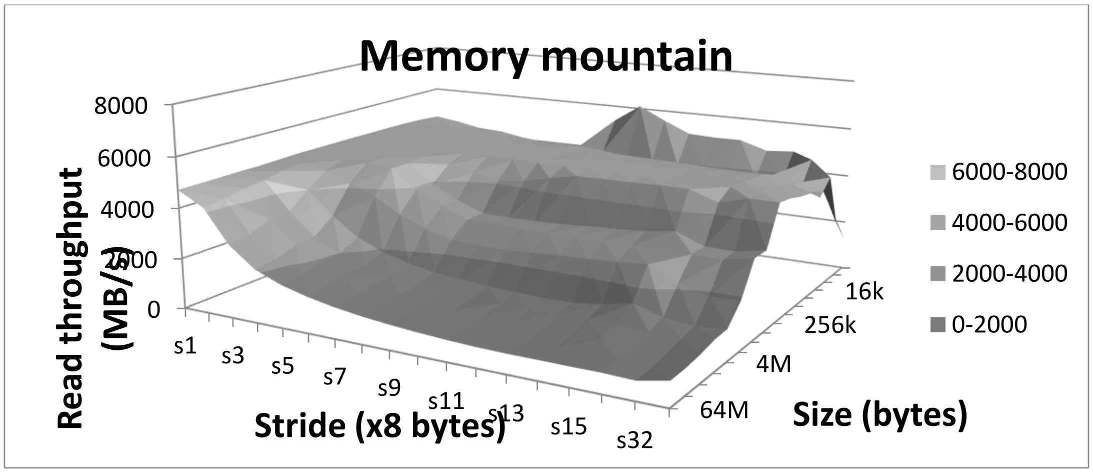
Compiled with -o3



Compiled with -o2



Compiled with -o1



When ‘mountain’ was compiled on my Linux VM the read throughput would increase by a few hundred, estimated, with each increased compile flag. The same pattern was exhibited when I compiled ‘mountain’ on my laptop running Linux. However, the increase was much smaller.

The memory mountain on my Linux VM does not resemble the memory mountain given in the textbook at all. It has a much sharper up turn at one sport rather than a gradual increase. I suspect this has to do with the fact that it is a VM and does not have a lot of cache allocated. The overall read throughput was much lower as well. The shape of the memory mountain on my laptop more resembles the memory mountain in the text book. However, the overall read throughput is significantly lower than the i7 Haswell. This is probably due to the fact that the laptop has an older i3.

From the memory mountains it’s hard to tell the separation between L1, L2, and L3 for the Linux VM. However, it is more clearly distinguished in the laptop. It looks like the i3 in the laptop has a L2 with size around 256k and L3 with size around 1-2 M.

2. Problem 7.8 from text

A. (a) main.1 (b) main.2

B. UNKNOWN

C. ERROR

3. Problem 7.12 from text

A. ADDR(S) = ADDR(.text) = 0x4004e0
ADDR(r.symbol) = ADDR(swap) = 0x4004f8

refaddr = ADDR(s) + r.offset
= 0x4004e0 + 0xa
= 0x4004ea

*refptr = (unsigned) (ADDR(r.symbol) + r.addend = refaddr)
= (unsigned) (0x4004f8 + (-4) - 0x4004ea)
= (unsigned) (0xa)
0xa

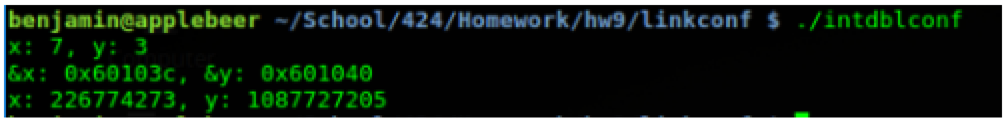
B. ADDR(S) = ADDR(.text) = 0x4004d0
ADDR(r.symbol) = ADDR(swap) = 0x400500

refaddr = ADDR(s) + r.offset
= 0x4004d0 + 0xa
= 0x4004da

*refptr = (unsigned) (ADDR(r.symbol) + r.addend = refaddr)
= (unsigned) (0x400500 + (-4) - 0x4004da)
= (unsigned) (0x22)
0x22

4. Problem 424-9

This is the output from the original source code.

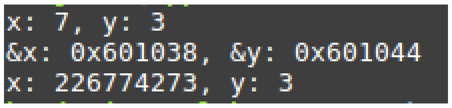


```
benjamin@applebeer ~/School/424/Homework/hw9/linkconf $ ./intdblconf
x: 7, y: 3
&x: 0x60103c, &y: 0x601040
x: 226774273, y: 1087727205
```

I had the same results when I ran it on my laptop and on my Linux VM. This bug is caused by the fact that p2() is called, it thinks that x is a double and assigns a double value to it which is too long and overflows into y.

I edited the line where x is declared in ***intdblconf2.c*** to ***double x = 12.0;.*** When I did this the linker threw an error that said ***multiple definition of 'x'***.

Then I edited the line where x is declared in ***intdblconf1.c*** to ***int x;***. When I did this there were the same warnings as the first time, but no error. This is the results:



```
x: 7, y: 3
&x: 0x601038, &y: 0x601044
x: 226774273, y: 3
```

Y is no longer being over written because the double declaration in ***intdblconf2.c*** is the strong symbol, so in main, x is a double. This way ***x*** is allocated large enough so that it can fit a double and ***y*** won't get overwritten.