Performance Lecture

Anna + Maria

Reminders

Congrats on finishing your assignments!

Why do we want to optimize performance?

1 px = 4 bytes

640px

512px

DEMO TIME

Making **gl_clear** really fast

How can we make this faster?

SPEED TIP #1

Reduce how many function calls you make!

FASTER!! FASTER!! FASTER!!
FASTER!! FASTER!! FASTER!!
FASTER!! FASTER!! FASTER!!
FASTER!! FASTER!! FASTER!!
FASTER!! FASTER!! FASTER!!
FASTER!! FASTER!! FASTER!!
FASTER!! FASTER!! FASTER!!
FASTER!! FASTER!! FASTER!!

Can we make this faster?

SPEED TIP #2

Use optimization option flags when you're compiling!

O1

```
-fauto-inc-dec
-fbranch-count-reg
-fcombine-stack-adjustments
-fcompare-elim
-fcprop-registers
-fdce
-fdefer-pop
-fdelayed-branch
-fdse
-fforward-propagate
-fguess-branch-probability
-fif-conversion
-fif-conversion2
-finline-functions-called-once
-fipa-profile
-fipa-pure-const
-fipa-reference
-fipa-reference-addressable
-fmerge-constants
-fmove-loop-invariants
-fomit-frame-pointer
-freorder-blocks
-fshrink-wrap
-fshrink-wrap-separate
-fsplit-wide-types
-fssa-backprop
-fssa-phiopt
-ftree-bit-ccp
-ftree-ccp
-ftree-ch
-ftree-coalesce-vars
-ftree-copy-prop
-ftree-dce
-ftree-dominator-opts
-ftree-dse
-ftree-forwprop
-ftree-fre
-ftree-phiprop
-ftree-pta
```

O2

```
-falign-functions  -falign-jumps
-falign-labels  -falign-loops
-fcaller-saves
-fcode-hoisting
-fcrossjumping
-fcse-follow-jumps  -fcse-skip-blocks
-fdelete-null-pointer-checks
-fdevirtualize  -fdevirtualize-speculatively
-fexpensive-optimizations
-ffinite-loops
-fgcse  -fgcse-lm
-fhoist-adjacent-loads
-finline-functions
-finline-small-functions
-findirect-inlining
-fipa-bit-cp  -fipa-cp  -fipa-icf
-fipa-ra  -fipa-sra  -fipa-vrp
-fisolate-erroneous-paths-dereference
-flra-remat
-foptimize-sibling-calls
-foptimize-strlen
-fpartial-inlining
-fpeephole2
-freorder-blocks-algorithm=stc
-freorder-blocks-and-partition  -freorder-functions
-frerun-cse-after-loop
-fschedule-insns  -fschedule-insns2
-fsched-interblock  -fsched-spec
-fstore-merging
-fstrict-aliasing
-fthread-jumps
-ftree-builtin-call-dce
-ftree-pre
-ftree-switch-conversion  -ftree-tail-merge
-ftree-vrp
```

O3

```
-fgcse-after-reload
-fipa-cp-clone
-floop-interchange
-floop-unroll-and-jam
-fpeel-loops
-fpredictive-commoning
-fsplit-loops
-fsplit-paths
-ftree-loop-distribution
-ftree-loop-vectorize
-ftree-partial-pre
-ftree-slp-vectorize
-funswitch-loops
-fvect-cost-model
-fvect-cost-model=dynamic
-fversion-loops-for-strides
```

Can we make this faster?

SPEED TIP #3

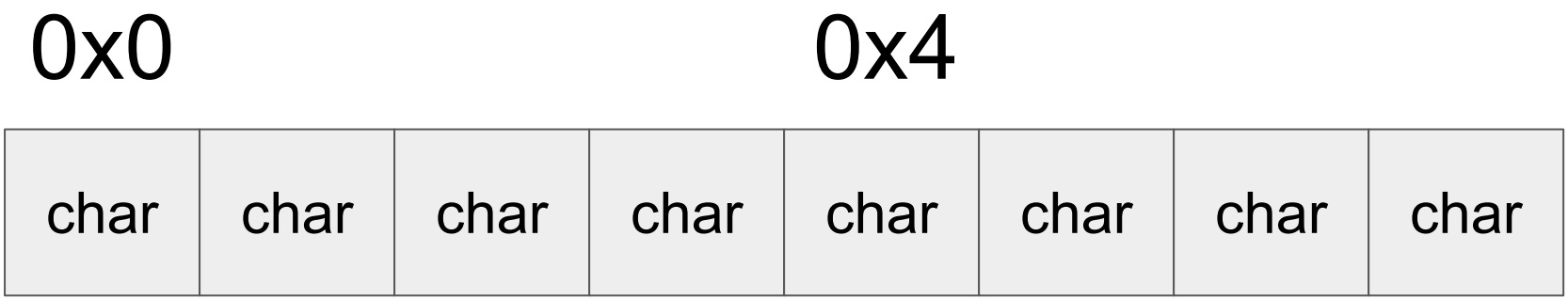
Precompute values you're planning to re-use!

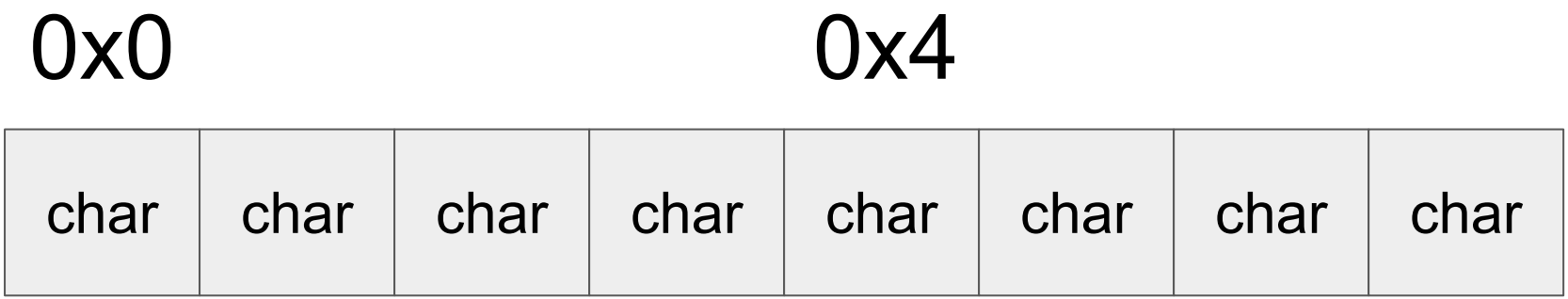
Can we make this faster?

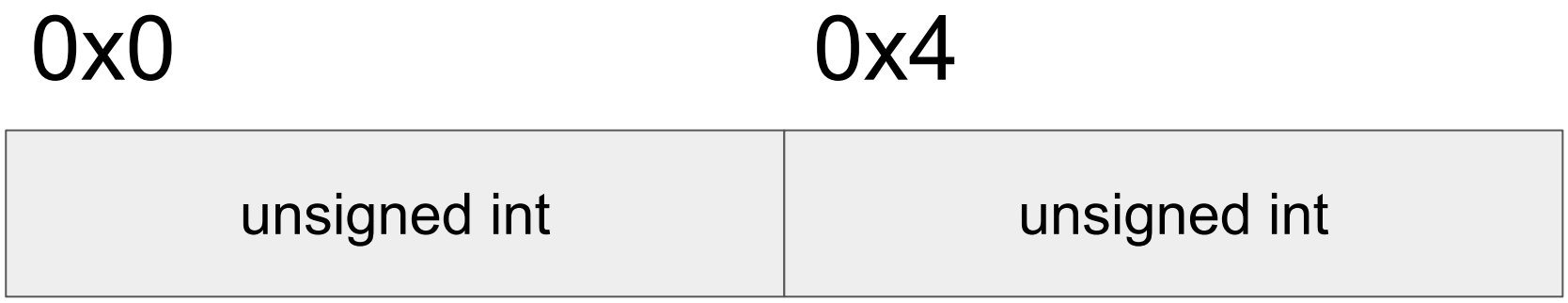
SPEED TIP #4

Only use **volatile** when
you really need to!

Can we make this faster?


0x0
0x4
char
char
char
char
char
char
char
char


0x0
0x4
char
char
char
char
char
char
char
char


0x0
0x4
unsigned int
unsigned int

SPEED TIP #5

Minimize the number of loads and stores you do! Work with more bytes at once!

Can we make this faster?

SPEED TIP #6

If you want to write 8 bytes at a time, you can use the **long long** data type!

Can we make this faster?

SPEED TIP #7

Unroll your for-loops >:)

Can we make this faster?

What's that we hear???

cache

cache

cache

cache

cache

cache

cache

cache

cache

SPEED TIP #8

Use the cache!

Can we make this faster?

SPEED TIP #9

The lower-level you get, the more you can MAXIMIZE your EFFICIENCY

1000x faster!!

Speed vs. style

Not everything should be treated as an optimization problem!



Good luck in week 10!