# Cortex-A15

# Cortex-A15

- ARMv7
  - Cortex-A15世代から整数除算命令が入った
- NVIDIA Tegra K1, 4+1-core, 2.1 GHz
  - DDR3L 4GB

# Chrome OS

- Linux + Glibcの割と普通の構成だった
- シェルにも簡単に降りられる
- デフォルトではコンパイラやヘッダは入っていない
  - /は書き換えない方が話が簡単そう
    - 自動アップデート機能との絡み
- OpenSSHやtar, xzは入ってる

# コンパイラ

- Chromium OS開発環境でクロスコンパイラ作成
- クロスコンパイラでターゲットホストのコンパイラ作成
  - これは動的リンクがうまく行かない（ありがち）
- ターゲットで上記コンパイラを使ってコンパイラをコンパイル
  - これを2回ほど繰り返すとCINT2006のVerification errorが出ないコンパイラができる（ありがち）

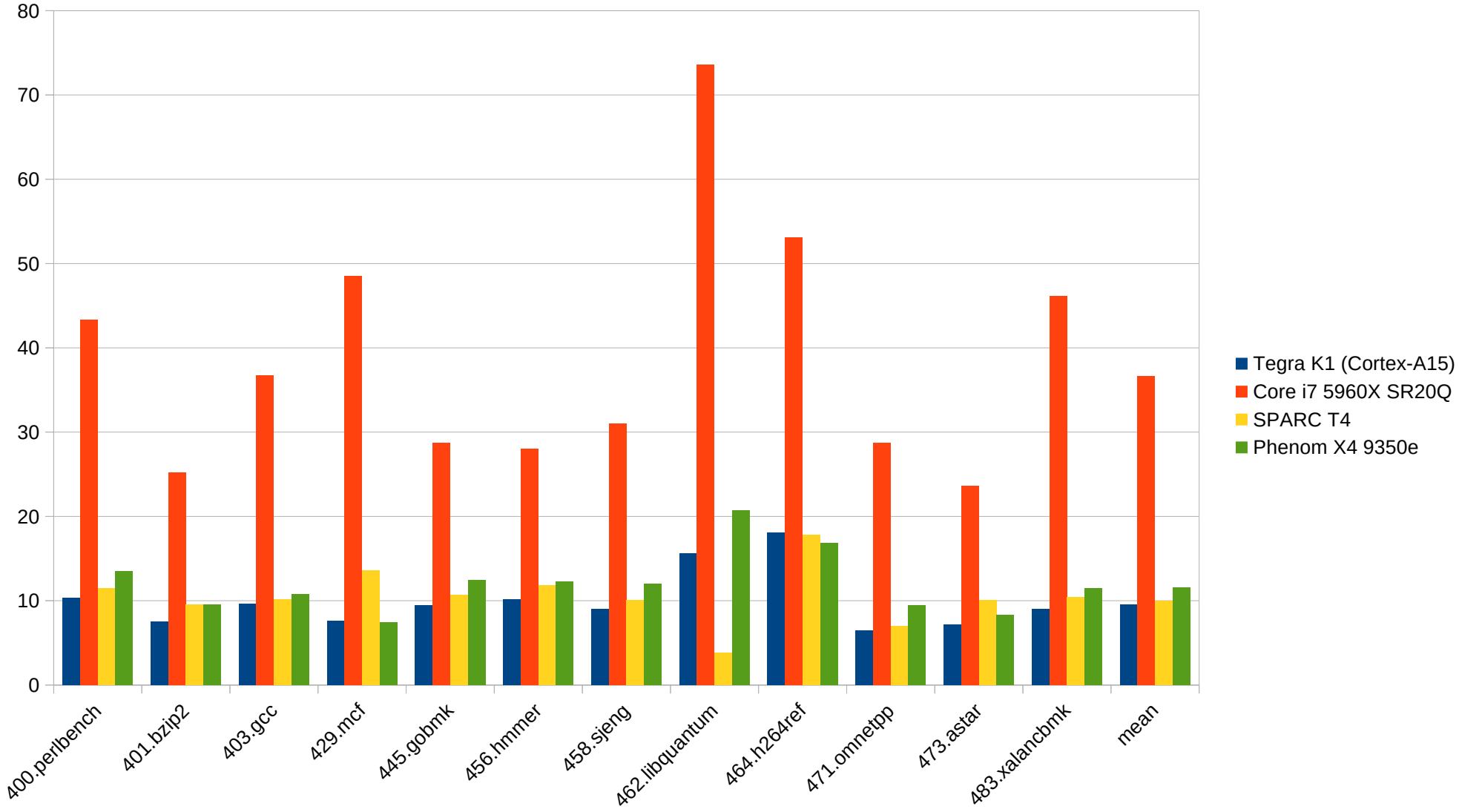
# 学び

- いくつかのLinux環境では/usr/lib/libc.so, /usr/lib/libpthread.soはELFファイルではなくリンカスクリプト
- ここにlibc.aやlibpthread.aのパスがハードコードされている

# 比較対象

| | |
|---|---|
| Tegra K1 (Cortex-A15) | 2100 |
| Core i7 5960X SR20Q | 3500 |
| SPARC T4 | 2848 |
| Phenom X4 9350e | 2000 |

# CINT2006 (完走)


80
70
60
50
40
30
20
10
0
400.perlbench
401.bzip2
403.gcc
429.mcf
445.gobmk
456.hmmer
458.sjeng
462.libquantum
464.h264ref
471.omnetpp
473.astar
483.xalancbmk
mean
Tegra K1 (Cortex-A15)
Core i7 5960X SR20Q
SPARC T4
Phenom X4 9350e

# ARM注意点

- charがunsigned char (-fsigned-char)
- NEON FPUはdenormal非対応（0扱い)

- ABIがたくさんある
- GCCの問題は多め
  - そもそもundefined behaviorを踏んでるコード多い
  - -fno-aggressive-loop-optimization
- x86/SPARCよりストレージモデルが緩い

# 結論

- 416.gamess, 482.sphinx3 (CFP2006)もミーティング終了頃には終わっているはず

- Chrome OSはLinux環境として使える
- Cortex-A15はCortex-A9の倍近く早くなっていた
  - Cortex-A57より少し遅いくらい
  - X-Gene1と同じくらい
  - ARMv8命令セットにしたことによる性能向上というわけではなかったらしい