# Posudek oponenta bakalářské práce

**Student:** Kratochvíl Radim, Bc.
**Téma:** Akcelerace grafických algoritmů pomocí koprocesoru NEON (id 17014)
**Oponent:** Jaroš Jiří, Ing., Ph.D., UPSY FIT VUT

**1. Náročnost zadání** **průměrně obtížné zadání**

Cílem bakalářské práce bylo experimentálně otestovat výkonnost koprocesoru NEON architektury ARM. K tomuto účelu bylo zvoleno několik jednoduchých benchmarků, jenž byly implementovány v C, assembleru a pomocí intrinsických instrukcí.
**Vhledem k nepřišil vysoké složitosti benchmarkových funkcí, hodnotím zadání jako průměrně obtížné.**

**2. Splnění požadavků zadání** **zadání splněno**

Zadání bylo splněno v plném rozsahu.

**3. Rozsah technické zprávy** **je v obvyklém rozmezí**

Technická zpráva je v obvyklém rozsahu.

**4. Prezentační úroveň předložené práce** **80 b. (B)**

**Prezentační úroveň je mírně nadprůměrná.** Práce je čtivá a dobře srozumitelná. Jednotlivé kapitoly dobře navazují a celkově se lze v práci snadno orientovat:

- Teoretická část shrnuje architekturu procesoru ARM a koprocesoru NEON. Není ovšem objasněno, proč byl zvolen právě procesor Cortex-A9 (a ne třeba novější A15).
- Dále práce popisuje několik zvolených algoritmů pro zpracování obrazu, např. převod do odstínů šedi, dolní a horní propust, Sobelův operátor a mediánový filtr. Zde je nutné podotknout, že zvolené algoritmy jsou poměrně jednoduché.
- Praktická část pak prezentuje dosažené výsledky. Zde bych vytkl popisný charakter na místo vysvětlujícího - tedy lepší zdůvodnění, proč jsou výsledky takové jaké jsou. Rovněž, srovnávat se vůči, C kódu bez optimalizace je velmi naivní.

**5. Formální úprava technické zprávy** **80 b. (B)**

Po formální stránce je práce lehce nadprůměrná, lze práci vytnout jen drobnosti.

- Některé obrázky jsou vysvětleny jen povrchně (obr 2.1, 2.2, 2.6 a 2.7). Obrázek 2.7 je navíc poměrně daleko od první citace.
- Pseudokódy mohly být lépe odděleny, např. rámečkem (listing v latexu).
- Popisky tabulek bych umístil spíše nad tabulku.
- Tabulky 5.1, 5.3, 5.5, 5.9 nemají vysvětleno, co se nachází v jednotlivých sloupcích.

**6. Práce s literaturou** **70 b. (C)**

Práce čerpá z 18 zdrojů, převážně z internetu (manuál ARM, popisy jednotlivých algoritmů). Důvěryhodnost některých zdrojů není příliš vysoká.
Ocenil bych nějakou knižní publikaci (např. na téma automatické vektorizace kódu), dále nějakou publikaci zabývající se vektorizací kódu pro platformy ARM, případně odkaz na výsledky dosažené jinými autory.

**7. Realizační výstup** **70 b. (C)**

Zdrojové kódy jsou vhodně členěné do modulů a funkcí. Bohužel zde není téměř žádný komentář (především C a intrinsic verze). Intrinsic a asm varianty vypadají rozumě optimalizované. O C verzi se to však říci nedá - jsou zde zbytečné operace, skoky které brání vektorizaci a daly by se eliminovat ternárním operátorem (kompilátor pak generuje instrukci blend).

**8. Využitelnost výsledků**

Bez srovnání s jinými autory a využití obvyklých výkonnostních metrik (FLOPS, cache miss) není možné objektivně posoudit kvalitu implementace. Přesto si myslím, že výsledky alespoň částečně ukazují možnosti koprocesoru NEON a daly by se využít při tvorbě složitějších programů. Jen pro informaci by bylo zajímavé, jak si s podobou úlohou pradí běžný x86 procesor.

**9. Otázky k obhajobě**

- Zkoušel jste upravovovat i větší obrázky, které se již nevejdou do cache? Na jaké velikosti obrázku jste vlastně testoval?
- Proč jste použil v C kódu 32b int, zatímco v asm kódu 8/16b?

- Jak daleko si myslíte, že jste od teoreticky maximálního výkonu koprocesoru NEON (efektivita implementace)?
- Uvažoval jste i o pokročilých knihovnách pro vektorizaci (OpenMP, vector classes, atd.)?
- Zkoumal jste nějaké další výkonnostní metriky, mimo času vykonání (FLOPS/FLIPS, cache miss, unit stall)?
- V textu se vyskytuje tvrzení: "kód napsaný v jazyce symbolických adres vykazuje stabilní urychlení výpočtu díky označení kódu jako volatile" (strana 30). Jak tomu mám rozumět? Jaký vliv má označení volatile na rychlost kódu?
- V textu tvrdíte, že intrinsic implementace byla pomalejší, z důvodu členění kódu do funkcí. Dále tvrdíte, že funkce byly inlinovány. Jak je tedy možné, že má členění do funkcí vlastnost vliv na rychlost?

## 10. Souhrnné hodnocení

**78 b. dobře (C)**

Prace prezentuje využití SIMD jednotky koprocesoru ARM Neon. Práce je poměrně dobře sepsána, dobře vysvětluje cíl práce a kroky provedené k jejímu dosažení. Autor zvolil několik poměrně jednoduchých algoritmů, kterými se snažil prokázat možnosti koprocesoru. Zde musím vytknout kvalitu C implementace (datový typ int, některé zbytečné operace) a rovněž porovnání výsledků, kde se jako reference bere C kód s vypnutou optimalizací (což se v praxi téměř nepoužije). Autor tak ukazuje zrychlení cca 60, zatímco reálně je pouze 3-4. Rovněž postrádám použití klasických metrik pro měření efektivity výpočtu (IPS, FLIPS, cache miss, propustnost paměti, atd.). **Proto práci hodnotím pouze průměrným hodnocením dobře (C).**

V Brně dne: 3. června 2015

.....
podpis