SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zavod za elektroniku, mikroelektroniku,
računalne i inteligentne sustave

Laboratorijska vježba 4

ARHITEKTURE RAČUNALA 2

GPGPU – OpenCL

Zagreb, 2018.

Predmet ove vježbe je ostvarivanje matričnog množenja u OpenCL-u. Razvit ćemo program koji će se moći izvršavati i na procesoru opće namjene i na grafičkom procesoru. Kako bismo provjerili točnost i korisnost razvijenog programa, dobivene rezultate i brzinu izvođenja usporedit ćemo s izravnom implementacijom u programskom jeziku C.

1 Priprema

OpenCL (Open Computing Language) je otvoreni standard koji omogućava pisanje usporednih programa koji se mogu izvoditi na različitim računskim uređajima [5, 3, 4, 2]. Podržani su procesori opće namjene (CPU), grafički procesori (GPU) i ostale vrste procesora i sklopovskih akceleratora (npr. DSP ili FPGA). OpenCL definira nadgradnje programskih jezika C i C++, te programska sučelja (API) za upravljanje izvođenjem programa na računskim uređajima.

Danas je posebno zanimljivo izvođenje programa na grafičkim procesorima [1]. Stoga će prvi korak biti instalacija izvršne okoline za GPU na kojem će se vježba izvršavati. Nastavak će ovisiti o konkretnom GPU uređaju kojeg imate na vašem laptopu te o operacijskom sustavu. Danas su najčešći integrirani Intelovi GPU-ovi, a pored njih prisutni su i proizvođači AMD i NVIDIA. Od operacijskih sustava najzastupljeniji su Windowsi, a prisutni su i Macovi i Linux. Svih $3 \times 3 = 9$ kombinacija trebalo bi raditi, a poteškoće se mogu očekivati ako imate novi GPU i stari OS ili obratno. Na laptopu s AMD-ovim GPU-om pod Ubuntuom potrebno je bilo napisati:

```
sudo apt-get install mesa-opencl-icd ocl-icd-opencl-dev clinfo
```

U okviru ove pripreme prvo je potrebno instalirati odgovarajuće izvršno okruženje za stroj na kojem ćete izraditi vježbu. Uspjeh instalacije potrebno je provjeriti prevođenjem minimalnog OpenCL programa koji je dostupan na adresi:

http://www.zemris.fer.hr/~ssegvic/ar2/lab4_cldemo.c

Minimalni program trebao bi se moći prevesti i GCC-om i MSVC-om. Za uspješno prevođenje programa na nekim će OS-ovima prevoditelju trebati dati informacije o kazalima koja sadrže zaglavlja `CL/cl.h` te binarnu biblioteku `OpenCL.so` (Linux, MacOSx) ili `OpenCL.lib` (Windows)[1].

Pitanja za vježbu:

1. Čemu služi `do { ... } while(0);` u definiciji makroa?
2. Čemu služi makro `CL_CHECK`?

[1] Ovdje su neke upute za postavljanje razvojne okoline koje po potrebi mogu biti proširene: https://gist.github.com/Ivan1248/962d32dc6a91eb9ac99b7742a3be5d28.

3. U kojem trenutku se pokreće prevođenje jezgrene procedure u OpenCL-u?
4. U kojem trenutku se pokreće izvođenje jezgrene procedure u OpenCL-u?
5. Kako OpenCL prevoditelj "zna" treba li jezgru (jezgrenu proceduru, engl. kernel function) prevesti za Intelov ili AMD-ov GPU?
6. Čemu služi poziv funkcije `get_global_id`? Izvodi li se ta funkcija na CPU-u ili na GPU-u? O čemu ovisi maksimalna povratna vrijednost?
7. Kojim funkcijama ostvarujemo prijenos podataka iz radne memorije u memoriju GPU-a i obratno?
8. Koje funkcije zauzimaju i otpuštaju memoriju GPU-a?
9. Koji parametar određuje koliko puta će se izvršiti jezgrena procedura?
10. Kako se provodi prebacivanje parametara iz programa u C-u u jezgrenu proceduru?
11. Na koji način zadajemo pokretanje jezgrene procedure?

2 Zadatci

1. Napišite naivnu implementaciju matričnog množenja u C-u. Neka dimenzije ulaznih matrica budu $n \times m$ i $m \times k$, pri čemu su n, m i k parametri.
2. Isprobajte vašu funkciju na malim matricama i ručno provjerite točnost rezultata.
3. Napišite izvedbu nove funkcije za množenje kvadratnih matrica koja izvođenje provodi na grafičkom procesoru (GPU) upotrebom odgovarajuće jezgrene funkcije. Provjerite točnost vaše izvedbe usporedbom s rezultatima prethodno razvijene izvedbe u C-u.
4. Iscrtajte grafove vremena izvođenja obje izvedbe na slučajno inicijaliziranim kvadratnim matricama 32-bitnih brojeva s pomičnim zarezom uz $n \in \{128, 256, 512, 1024, 2048\}$. Mjereno vrijeme treba uključivati prijenos matrica iz RAM-a u GPU memoriju i natrag, ali ne i prevođenje jezgrene procedure.
5. Profilirajte vrijeme izvođenja izvedbe za grafički procesor po odsječcima: prijenos RAM $\rightarrow$ GPU, računanje rezultata, prijenos GPU $\rightarrow$ RAM.
6. Procijenite brzinu prijenosa RAM $\rightarrow$ GPU i GPU $\rightarrow$ RAM u GB/s te ostvarenu brzinu računanja na GPU-u (TFLOPS). Usporedite dobivene veličine s brzinom prijenosa RAM $\rightarrow$ RAM te ostvarenom brzinom računanja na CPU-u.

7. Procijenite koji bi postupak bio prikladniji za izvedbu na GPU-u: postupak X sa složenošću O(N) ili postupak Y sa složenošću O(N^3).

References

[1] General-purpose computing on graphics processing units — Wikipedia, the free encyclopedia. 2018. URL https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units.

[2] Lee (https://stackoverflow.com/users/4127147/lee). OpenCL: work-group concept. 2018. URL https://stackoverflow.com/questions/26804153/opencl-work-group-concept.

[3] Matthew Scarpino. A gentle introduction to OpenCL — Dr. Dobb's. 2011. URL https://freecontent.manning.com/wp-content/uploads/a-gentle-introduction-to-opencl.pdf.

[4] Texas Instruments. Understanding Kernels, Work-groups and Work-items. 2018. URL http://downloads.ti.com/mctools/esd/docs/opencl/execution/kernels-workgroups-workitems.html.

[5] Jonathan Tompson. An introduction to the OpenCL programming model. 2012.