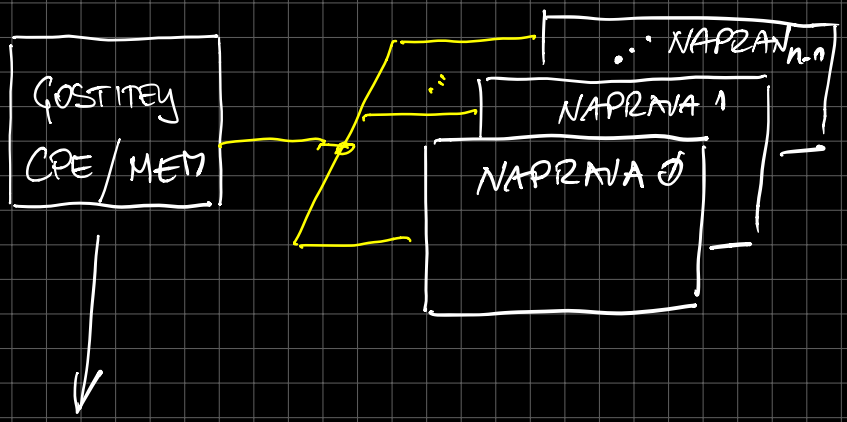
Delavnica "Programiranje CPE"

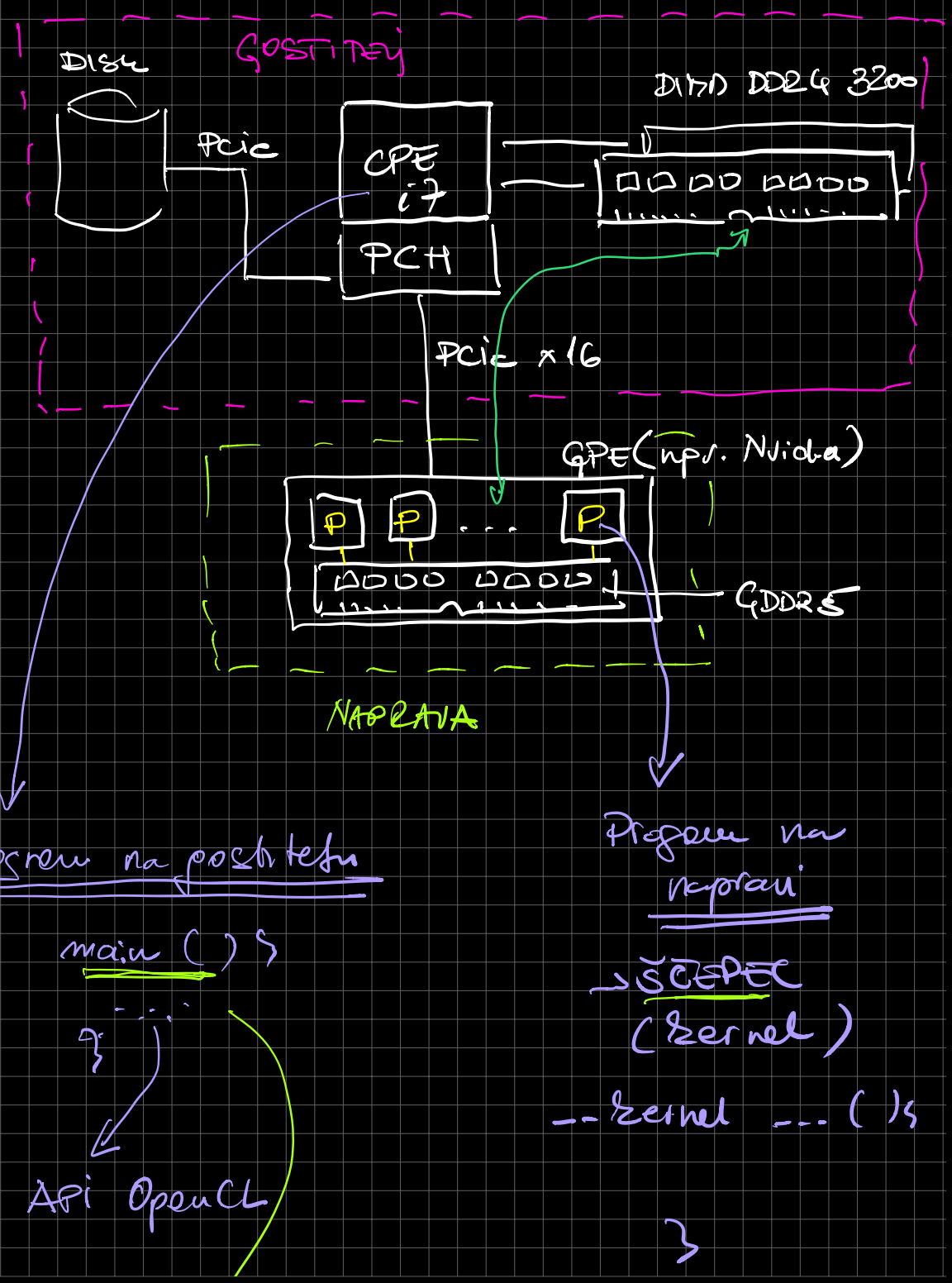
20.4.2021

MET. RAČ. SISTEM



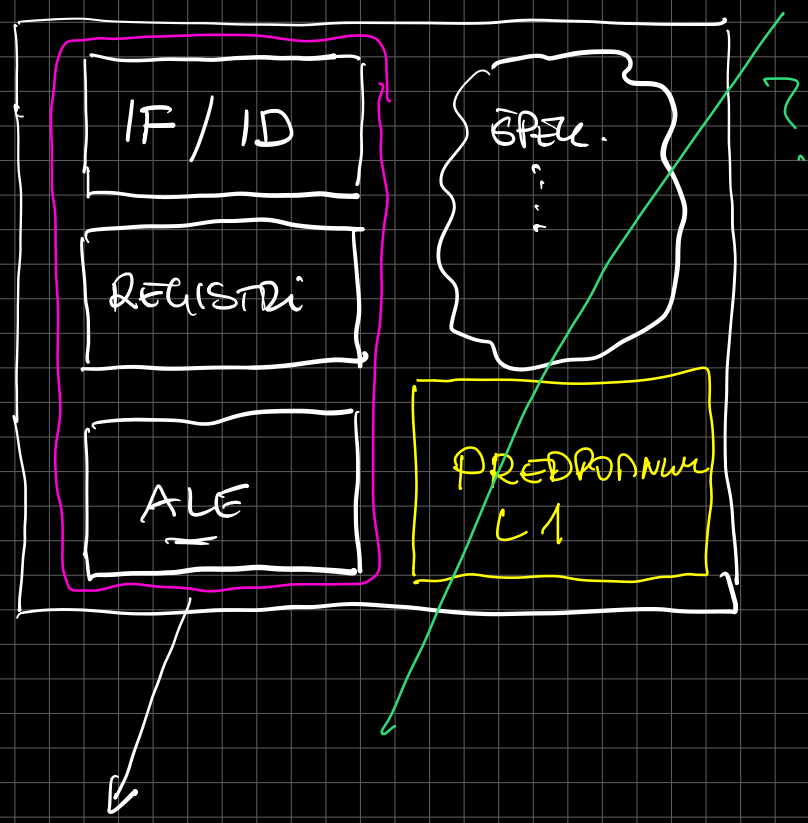
naloga:
- → pripravi vse podatke za naprave
- → pripravi programe, ki se bodo izvajali na napravah
- → prenos podatkov med enim MET in poljubno napravo
- → na naprave prenese program in ga požene

Primer:


GOSTITELJ
DISK
PCIe
CPE i7
PCH
DIMM DDR4 3200
PCIe x16
GPE (npr. Nvidia)
P
P
P
GDDR5
NAPRAVA
Program na gostitelju
- main ( ) {
}
API OpenCL
Program na napravi
→ ŠČEPEC (kernel)
--kernel ... ( ) {
}

Izvajanje navodil se
prekrivaji ni tako ščepec na
popravi

CPE

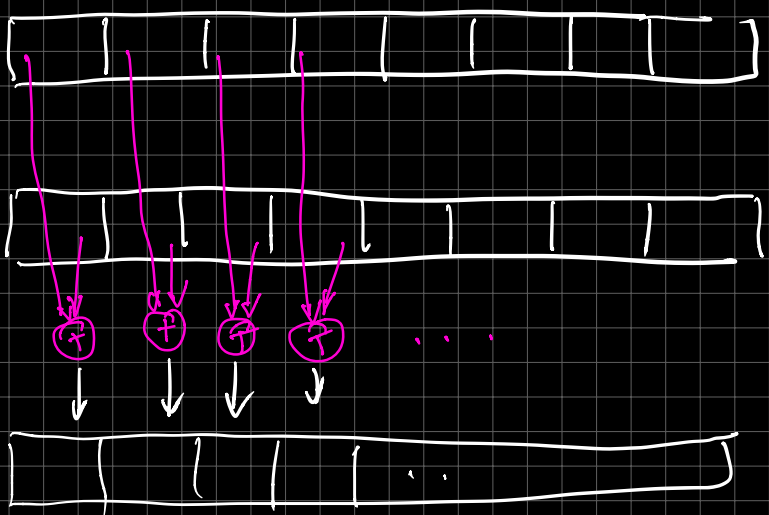



```
while (tid < 128) {
    c[tid] = a[tid] + b[tid];
    tid += 1;
}
```

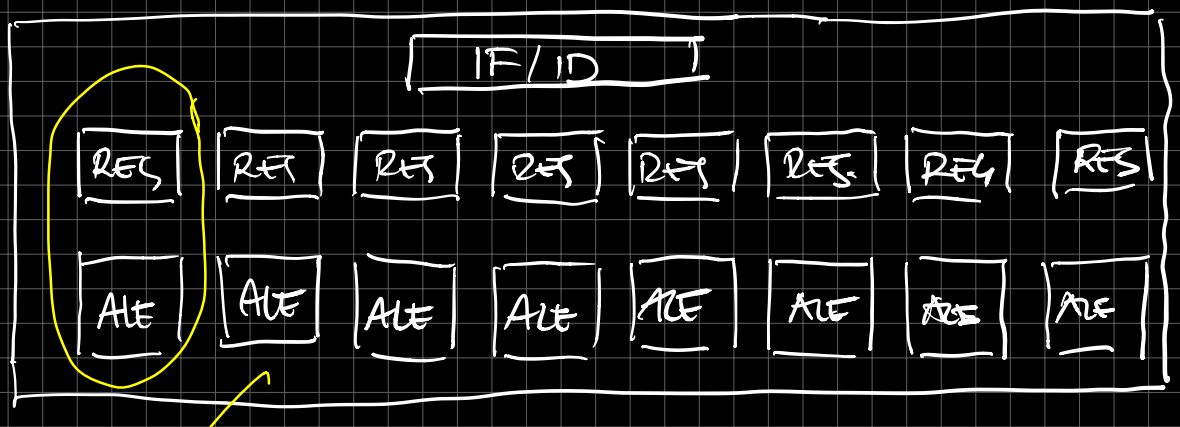
1. evolucija: Cray → SIMD → vektorske instrukcije

Vektorske instrukcije:

vadd:



potrebno:
- 8 ALE
- 8x registrski niz



```
while (tid < 128) {
    c[tid : tid+7] =
        a[tid : tid+7] + b[tid : tid+7];
    tid += 8;
}
```

VSEH 8 ALE DOBI ISTOČASNO ISTO OPARACIJO, ki jo DOLAJO IZVESTI

2. vaja: Zgodovina evolucije

```
c[tid] = a[tid] + b[tid];
```

22 22

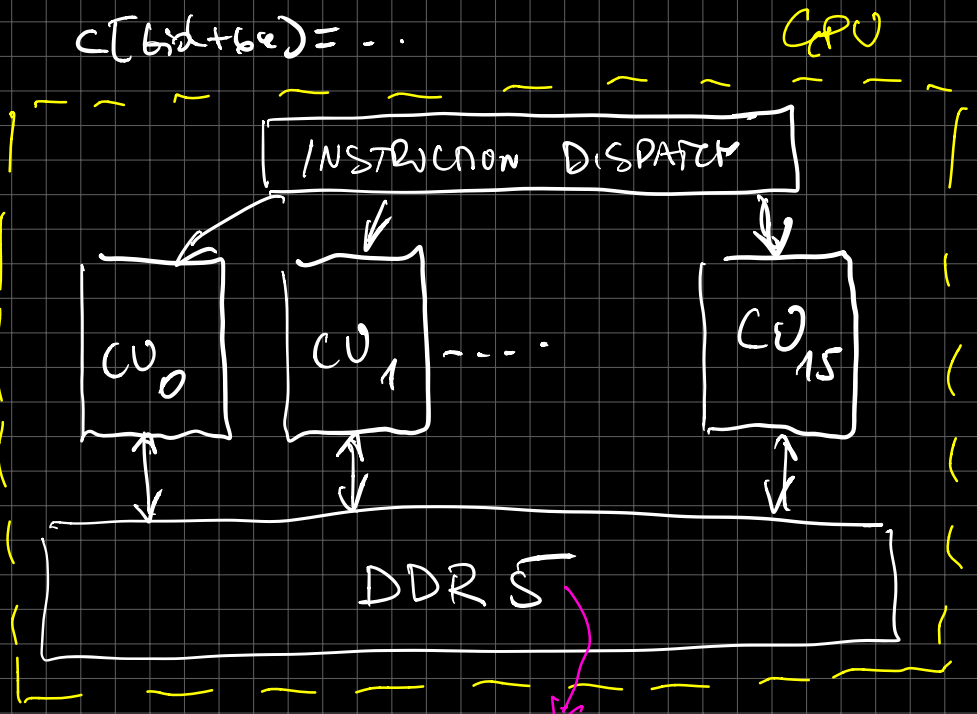
PRED: 22

load reg1, m[tid]
load reg2,
add reg1,reg2

Ča je CU jeve le 8 (= 64 PE)

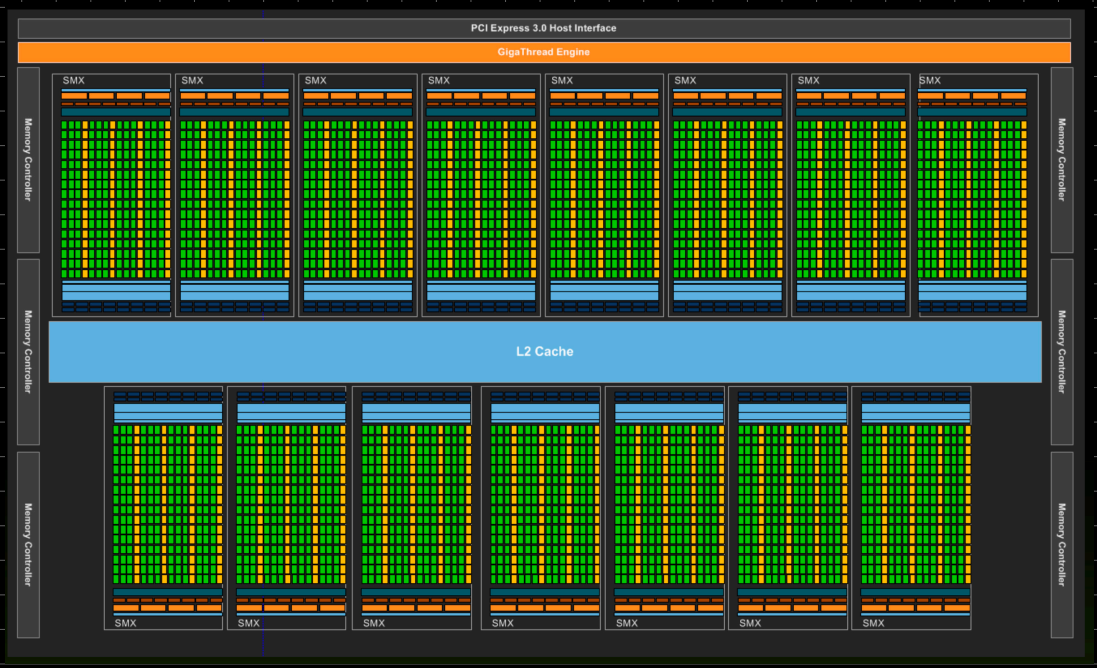
c[tid] = a[tid] + b[tid]

c[tid+be] = . .



NVIDIA TESLA K40 :

CU — Streaming Multiprocessor



OpenCL ABSTRAKCIJA HW

WORK-ITEM (NIT) ⟺ PE

WORK-GROUP (DELOVNA SKUPINA) ⟺ CU

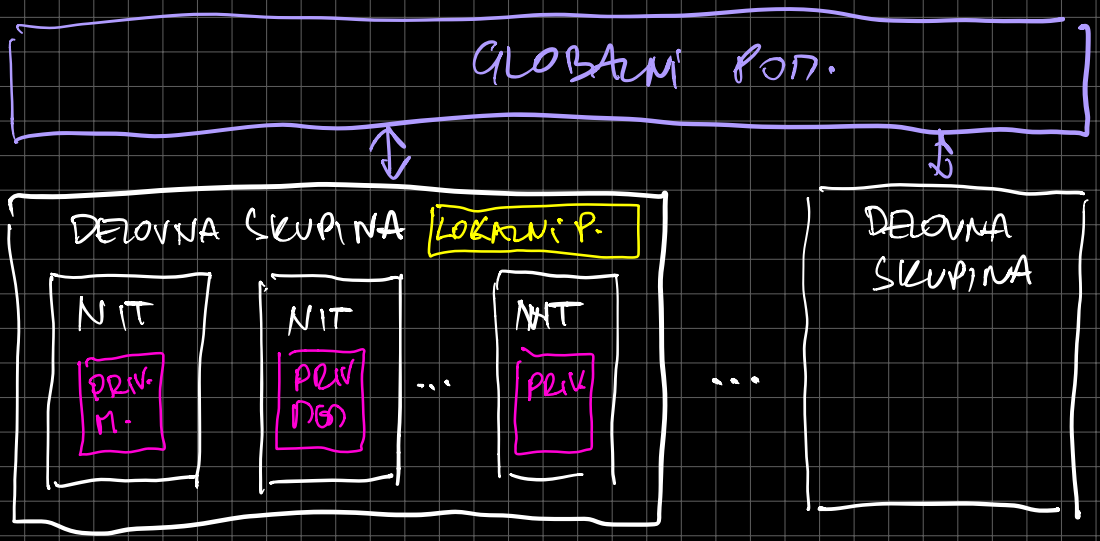
ABSTRAKCIJA PROCESNIH ENOT

- Vsebuje niti, ki se izvajajo na eni CU

GLOBAL MEMORY ⟺ GLOBALNI DDR5 POMNILNIK

LOCAL MEMORY ⟺ LOKALNI SRAM POMNILNIK

PRIVAT MEMORY ⟺ REGISTRI



OpenCL

OpenCL API

→ knjižnica funkcij, ki jih uporabljaš programom na gostitelju

OpenCL C

prog. jezik C različnega pisanja ščepcev

OpenCL okolje

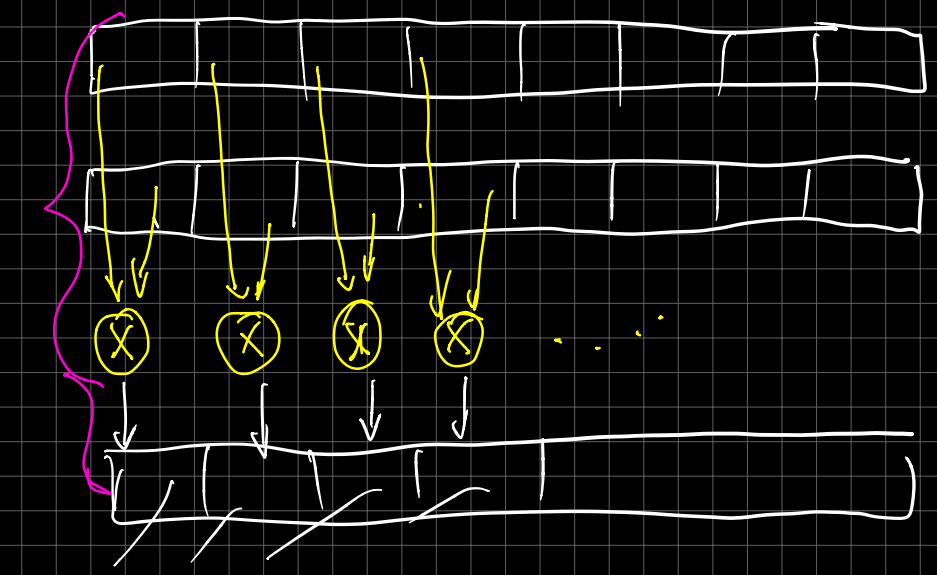
→ predem gostitelj prevede ščepce za napravo, mora ustvariti okolje:

1. gonilnik ⇒ PLATFORMA (npr. Nvidia ali AMD)
2. Naprave

⇒ TESNA USTVARJENA KONTEKST

→ naprave, programe, vrste ... ipd. določeš

PRVI PROGRAM NA GPE:

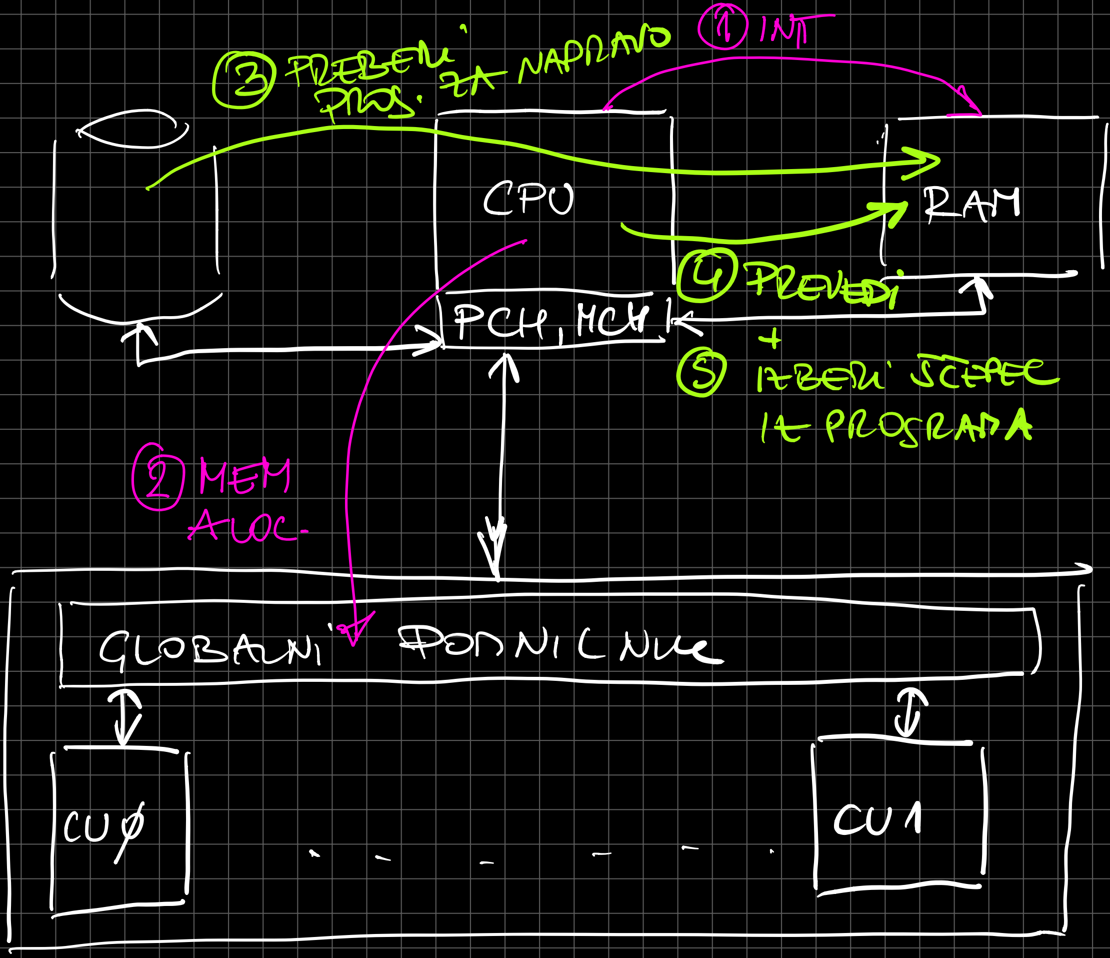


Σ

Šilpec → naredbo za eno samo nit

```
c[tid] = a[tid] * b[tid];
```

21.4. 2021

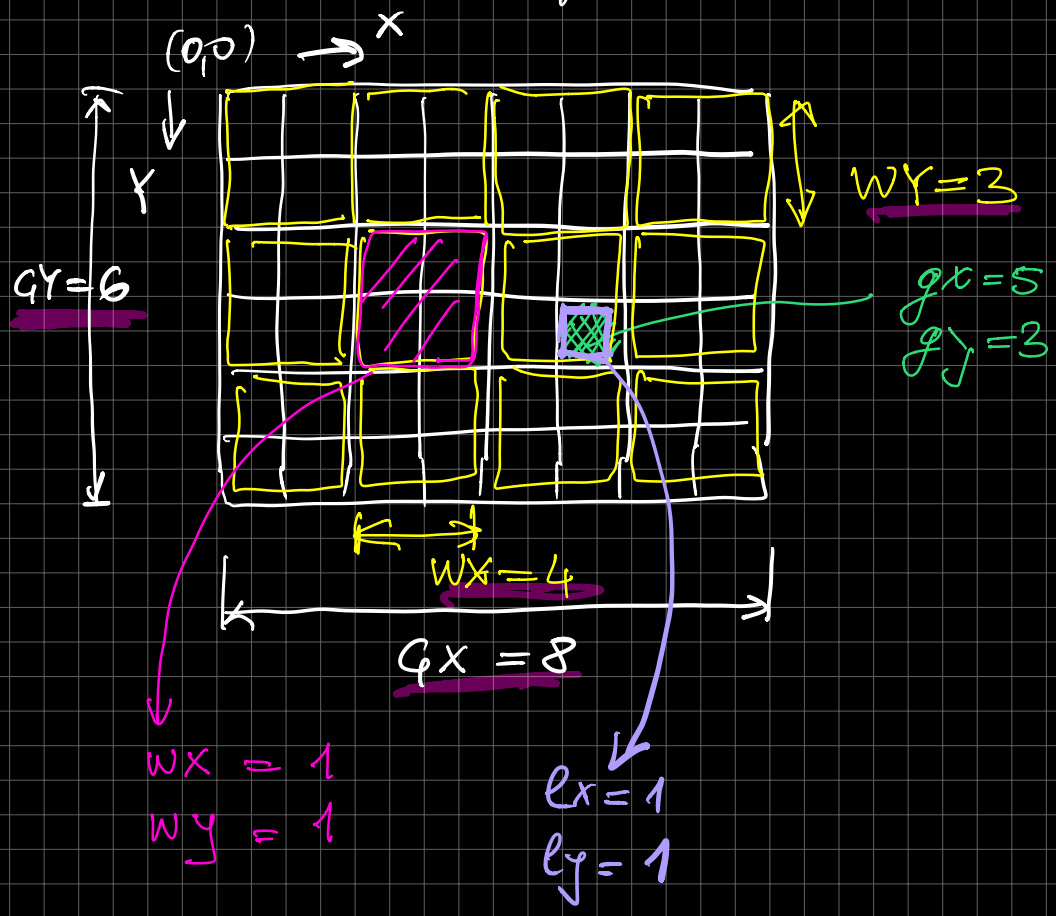


* NDRange

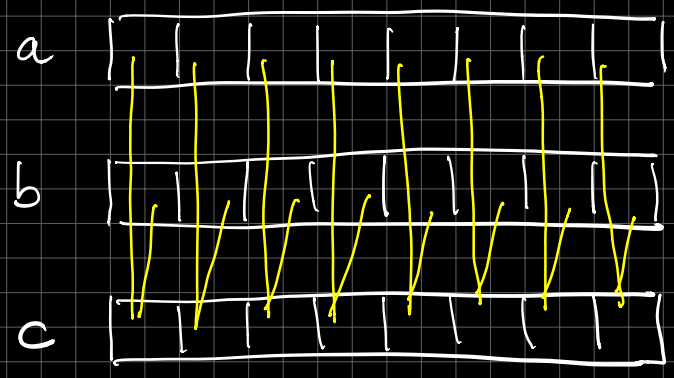
a) določa število in organizacijo po skupinah

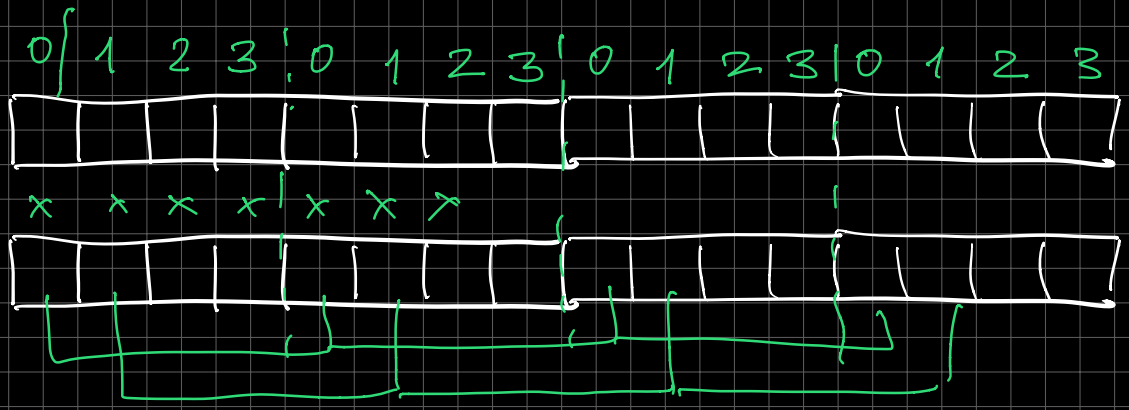
b) določa prostorsko organizacijo niti → 1D, 2D, 3D

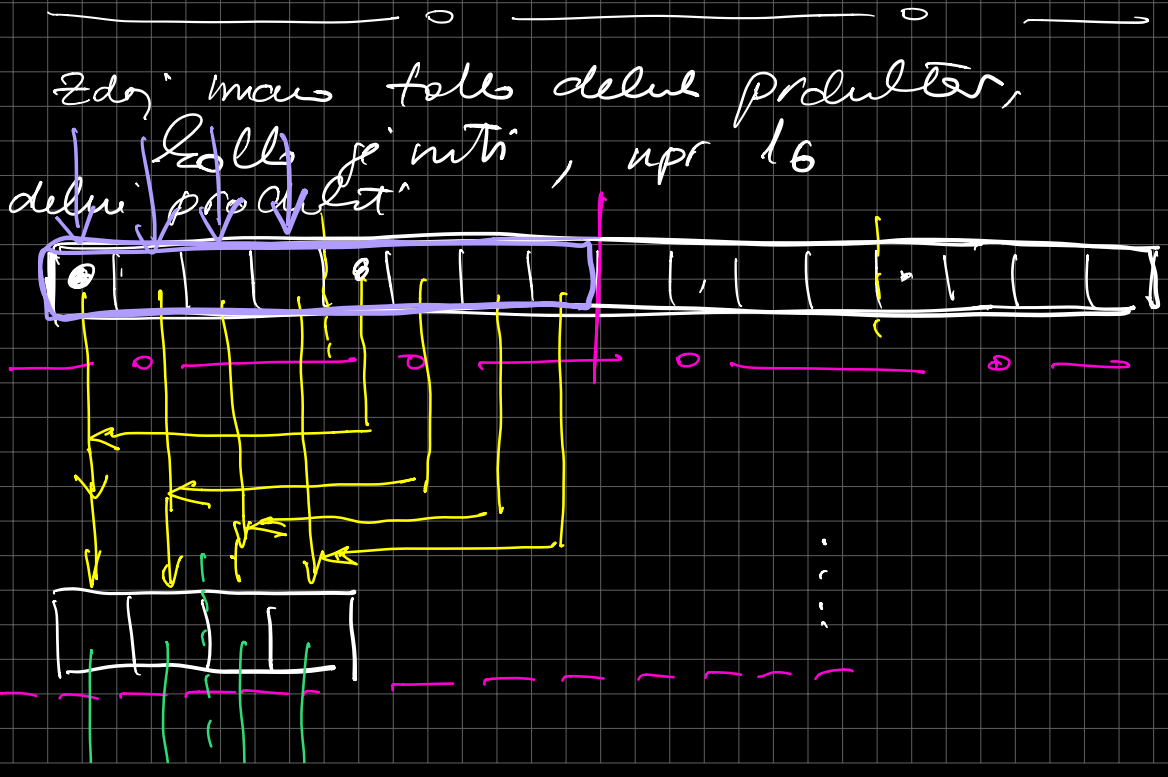
→ svaka nit ima svoj jedinstven dvodijelni index (ID) u prostoru NDRange

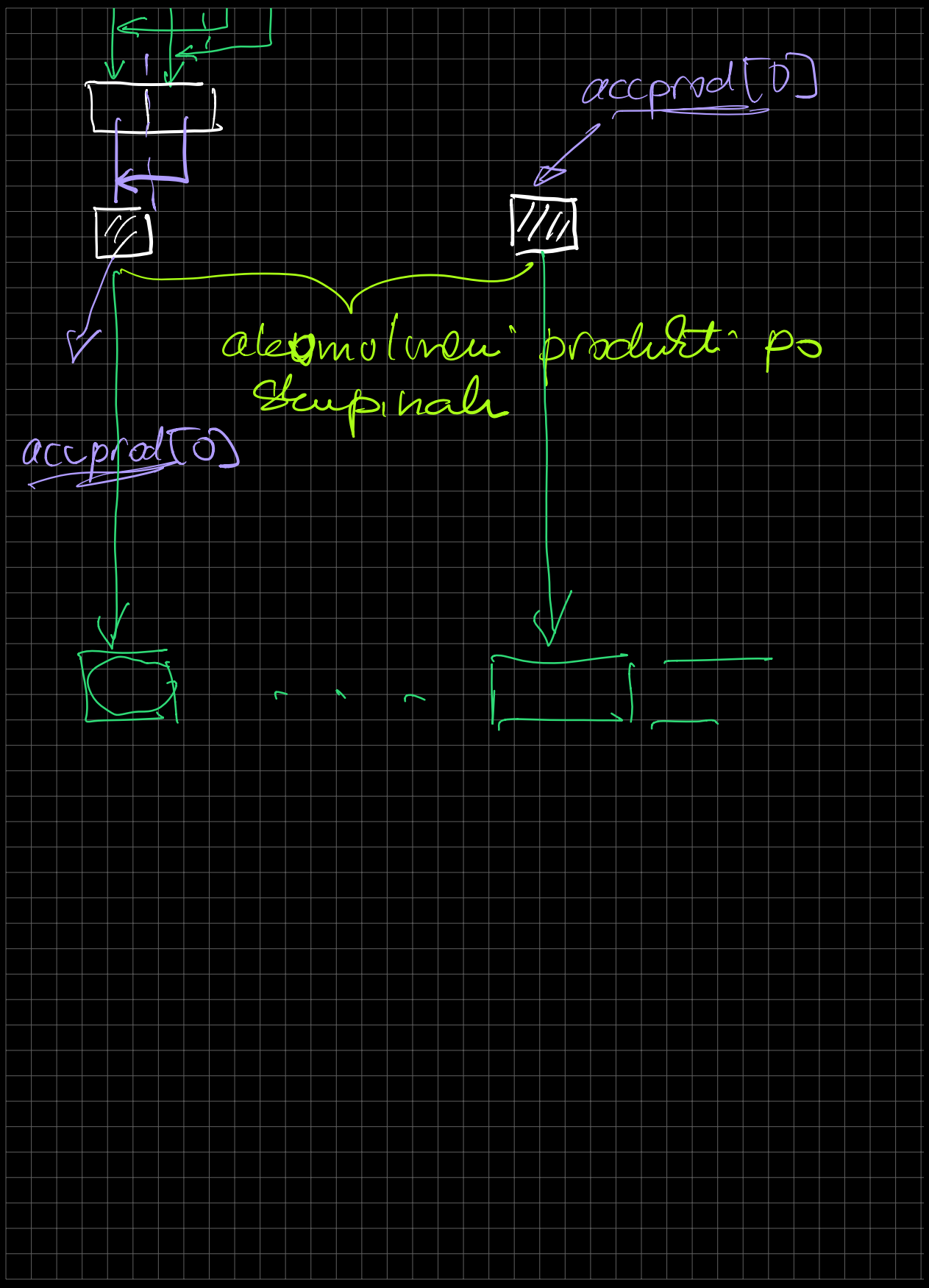


Predan računalu šaljemo, moramo definirati {GX, GY} i {WX, WY}


a
b
c


0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3


Ecco i numeri dello produttore,
Ecco le unità, cioè 16
dello prodotto


accprod[0]
akumulacni produkt po skupinah
accprod[0]

22.04.2021

Razvrščanje niti

→ GPE ne razporeja posameznih niti

CU

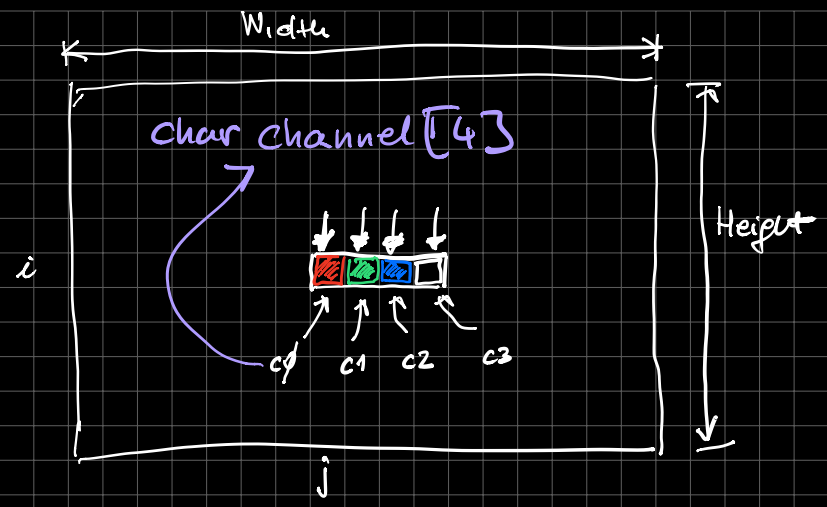
→ CU razporeja t.i. SNOPE NITI (WARPS)

def: množica 32 niti, katerih ukazi so vzporedni

Snop0: 0–31
Snop1: 32–63
⋮

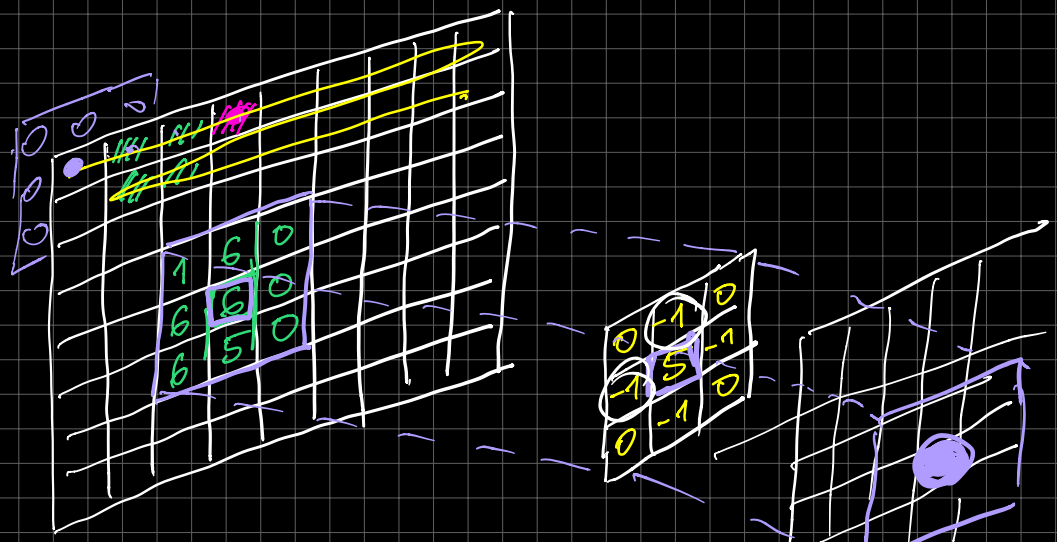
→ Na snope je vedno dostop do globalnega pomnilnika

V globalni pomnilnik se v nekem trenutku gre s naslovom, ki ga izstavijo niti iz istega snopa



$$pImage[(i * Width + j) \times 4] + channel$$

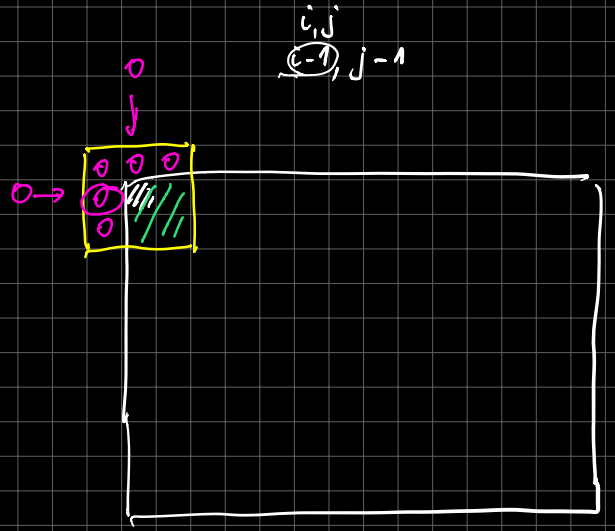
Filtrowanie: podkreślanie kolorów (Edge Enhancement)

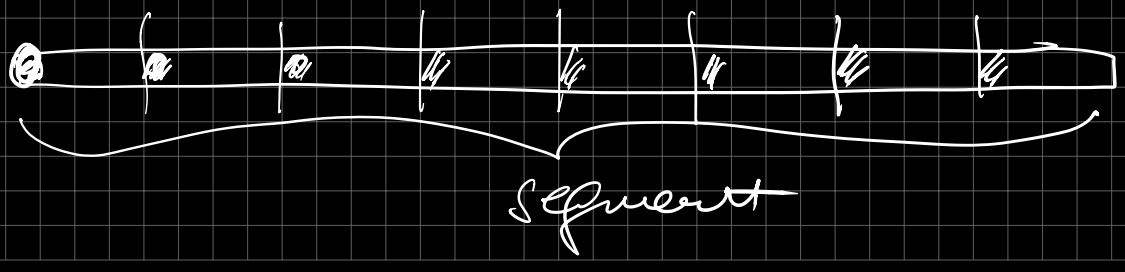


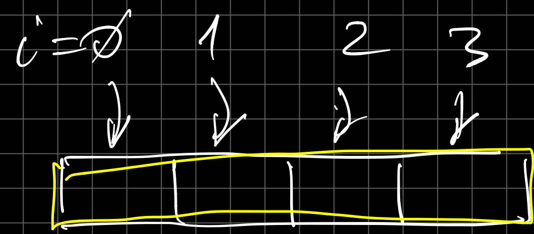
nowy pixel = $1 \cdot 0 + 6 \cdot (-1) + 0 \cdot 0 + 6 \cdot (-1) + 6 \cdot (5) \ldots$

Šcepec :

$$imageOut[i,j] = imageIn[i,j] * 5 - imageIn[i,j-1] - imageIn[i-1,j] - imageIn[i,j+1] - imageIn[i+1,j]$$







for (i = 0 ... 3) {

– beri iz p.b.

– filtriraj

}

1. Preberi vse 4 kanale naenkrat (4B)
2. for (0 ... 3) {

→ filtriraj vse prebrane bajte posebej

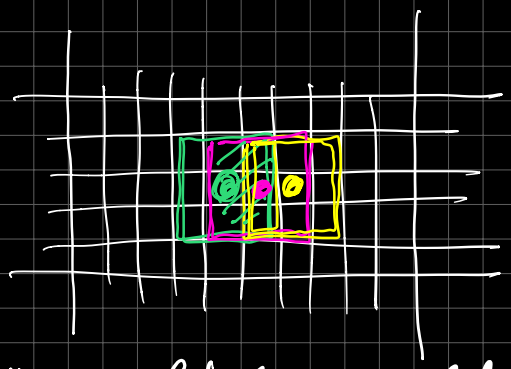
}

Izkoristili bomo:

→ vektorske pod. tip

i:

→ vektorske operacije



Uporaba lokalnega pomnilnika: