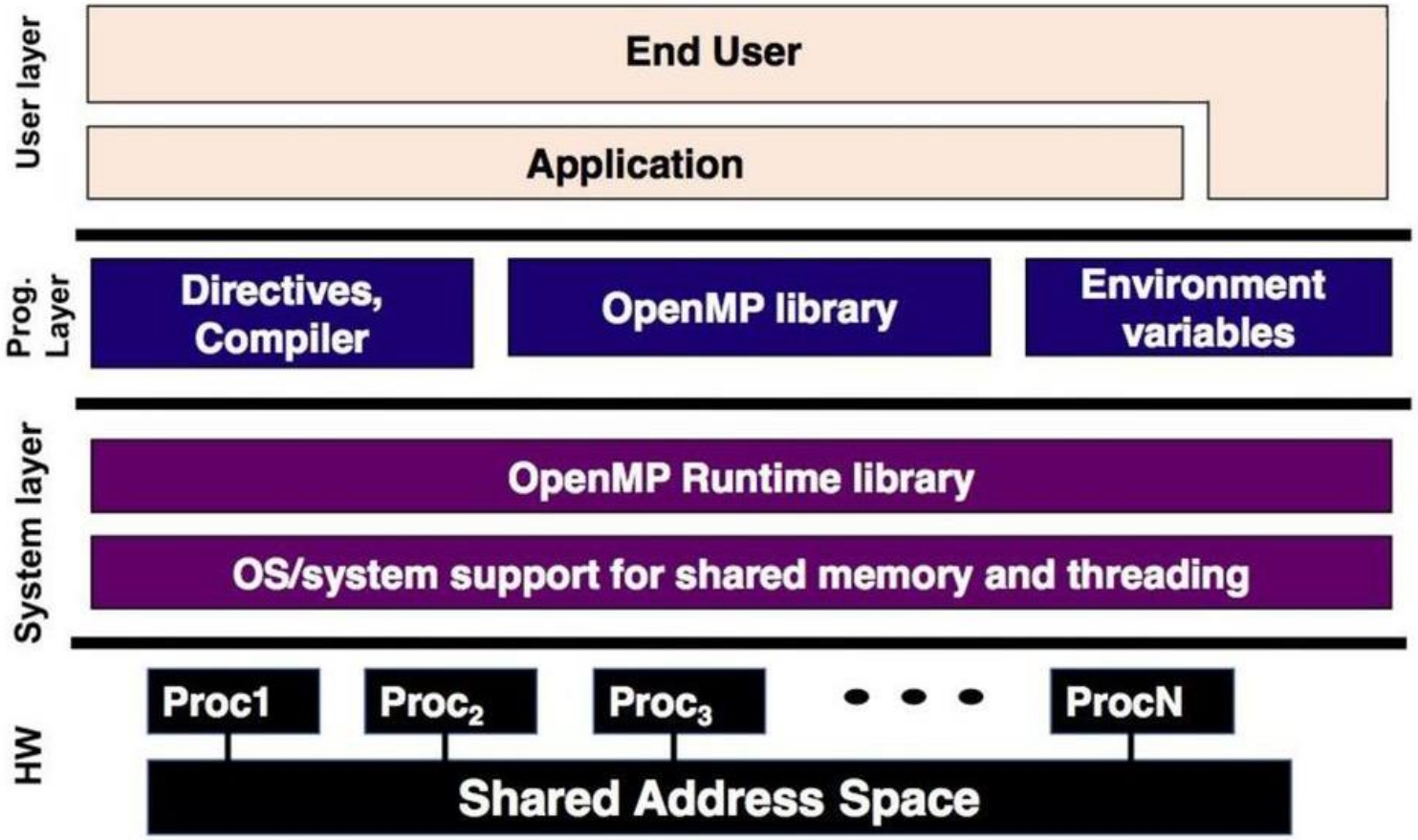
Paralelno programiranje sa OpenMP

Uvod

- **Open Multi-Processing** (OpenMP) je **API** (engl. *Application Programming Interface*) koji podržava **multi-platformsko paralelno programiranje** za **sisteme sa deljenom memorijom** u programskim jezicima **C**, **C++** i **Fortran**
- OpenMP je baziran na **portabilnom i skalabilnom modelu** koji pruža **inkrementalni razvoj** i **fleksibilan interfejs** za rad paralelnih programa, od desktop računara sa jednim višejezgarnim CPU do superračunara
- Sastoji se od:
  - **skupa direktiva programskom prevodiocu** (engl. *compiler directives*)
  - **bibliotečkih rutina** (engl. *library routines*)
  - **promenljivih okruženja** (engl. *environment variables*)

API



Izvor: F. Desprez, INRIA, https://slideplayer.com/slide/13088921/

Specifikacija

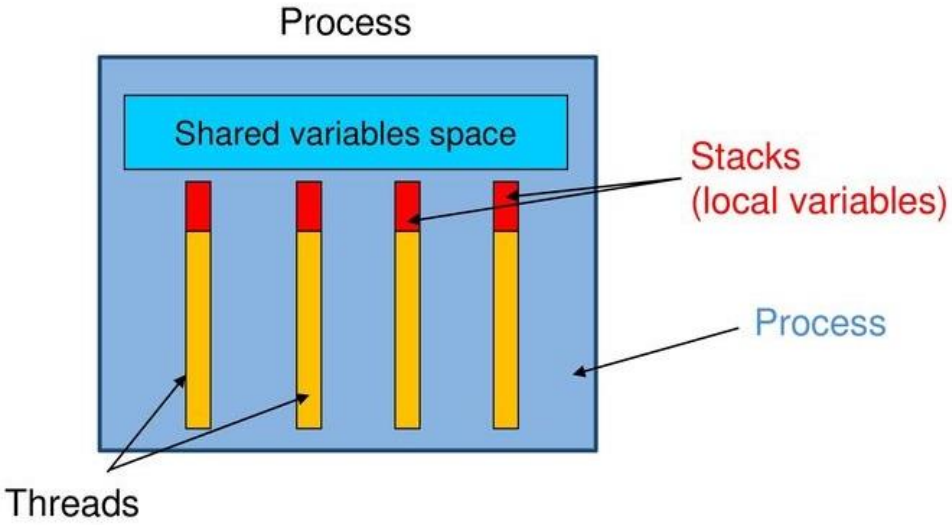
- Razvojem upravlja **OpenMP Architecture Review Board** (ARB), čiji su članovi Intel, AMD, IBM, Oracle, HP, Cray, …
- Prva verzija (1.0) za Fortran predstavljena u oktobru 1997, za C i C++ u oktobru 1998.
- Specifikacija javno dostupna na: https://www.openmp.org/specifications/
- **API** pruža **sloj apstrakcije visokog nivoa** nad **višenitnim** (engl. *multithreading*) **primitivama nižeg nivoa** koje se koriste u odgovarajućim jezicima

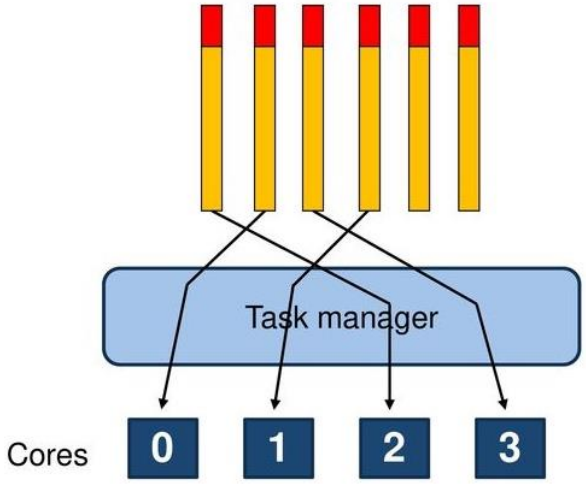
Osnove

- **Osnovna filozofija: proširenje sekvencijalnog programskog koda** primenom posebnih **direktiva programskom prevodiocu u stilu komentara** – tzv. **pragma** (skraćeno od engl. *pragmatic information*) **direktive** koje pružaju **programskom prevodiocu sugestije** kako može **paralelizovati kod**
- Ako prevodilac **ne podržava odgovarajuću opciju**, prosto će **ignorisati pragmu**
- Jednostavnim **dodavanjem pragma direktiva**, omogućava **paralelizaciju postojećeg serijskog koda**

Model

- **Višenitni** (engl. *multithreading*) pristup **paralelizaciji**
- **Procesi** i **niti**
  - **Niti** dele **resurse** unutar **procesa**, svaka nit ima svoj stek, SP i IP
  - **Planer zadataka** u okviru OS **dodeljuje zadatke** tj. niti pojedinim **jezgrima**

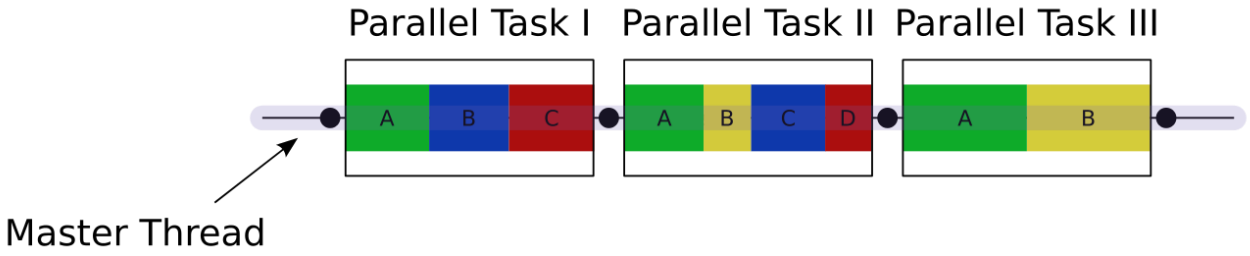


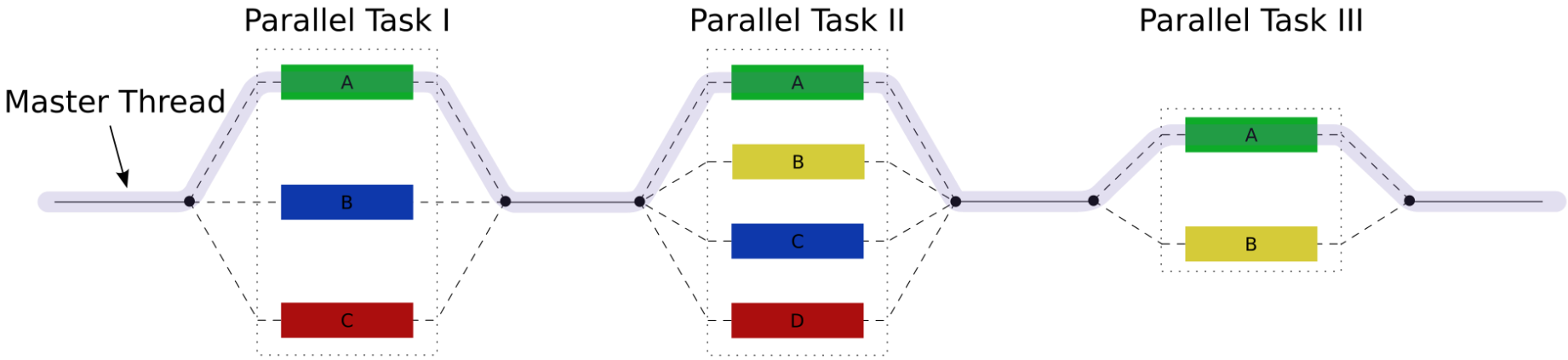


Izvor: F. Desprez, INRIA, https://slideplayer.com/slide/1308892 1/

Model

- **Paralelizam** tipa **račvanje-spajanje** (fork-join), **master nit** se izvršava **sekvencijalno** dok se ne naiđe na **paralelnu regiju**
  - Na kraju paralelne regije, sinhronizacija i terminacija svih niti osim master niti





Izvor: https://en.wikipedia.org/wiki/OpenMP

Sintaksa

- Neophodno da programski **prevodilac** ima **podršku za OpenMP**, prilikom prevođenja sa G++ potrebno dodati fleg `-fopenmp`
- Prototipovi funkcija i makro definicije se nalaze u posebnom zaglavlju:

```
#include <omp.h>
```

- Najveći broj konstrukcija kod OpenMP su **direktive programskom prevodiocu**:

```
#pragma omp konstrukcija [klauzula [klauzula] … ]
```

```
#pragma omp parallel num_threads(8)
```

- Najveći deo OpenMP konstrukcija se primenjuje nad **strukturiranim blokom** – blokom od jedne ili više naredbi sa tačno jednom ulaznom tačkom na vrhu i jednom izlaznom tačkom na dnu, zabranjene naredbe bezuslovnog skoka ali ne i poziv `exit()`

Tok paralelizacije

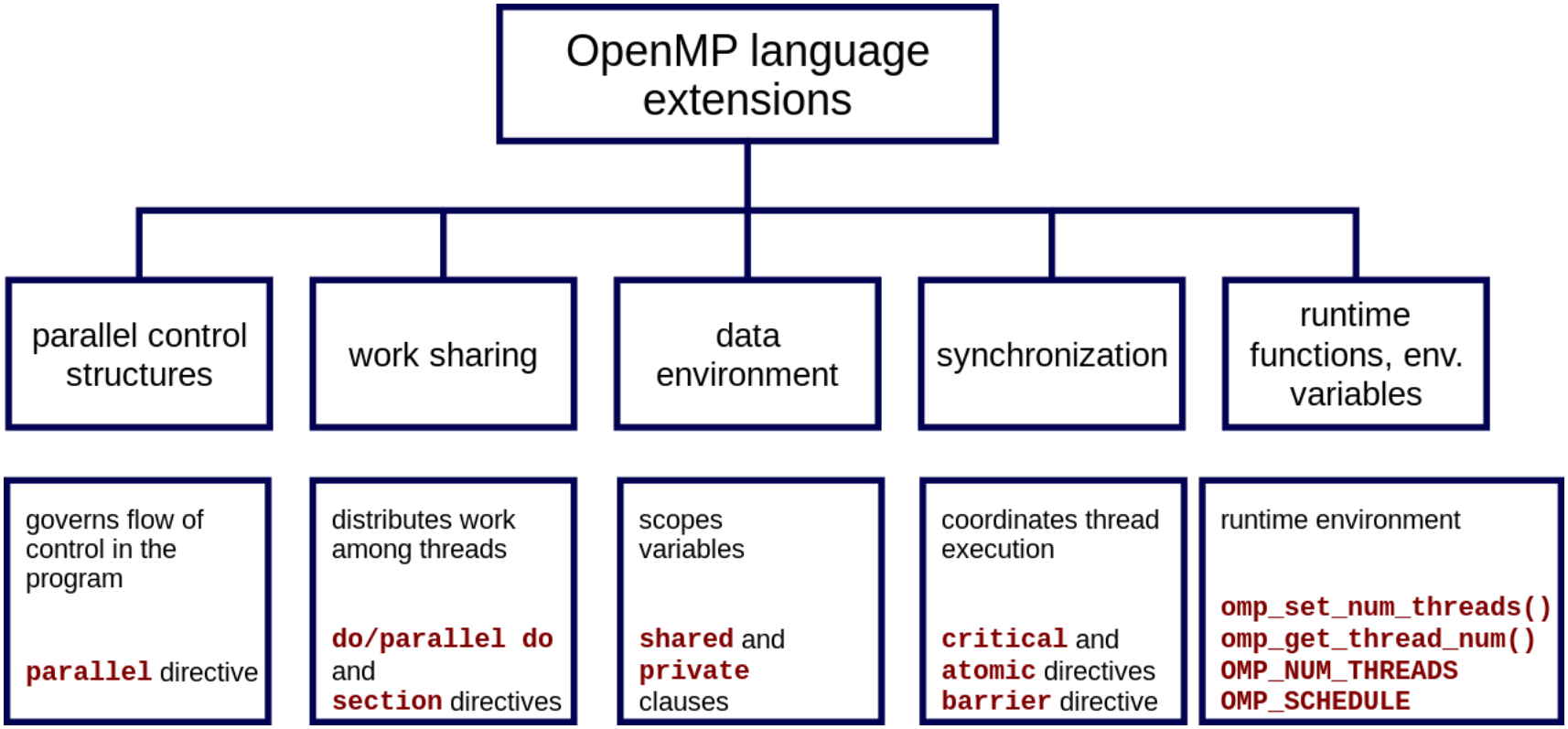
- Postupak paralelizacije primenom OpenMP svodi se realizaciju sledeća tri koraka:

1. **Polazna tačka** je **serijski program**

2. Analiziraju se postojeće **zavisnosti po podacima i uslovi trke**, kao i strukturirani blokovi koji mogu predstavljati **paralelne regije**

3. Programski kod se **proširuje OpenMP pragma direktivama** na odgovarajućim mestima

Konstrukcije



- Evolucija OpenMP konstrukcija: https://youtu.be/W2Zcrhe1-qg

Izvor: https://en.wikipedia.org/wiki/OpenMP

Primeri OpenMP paralelnih programa

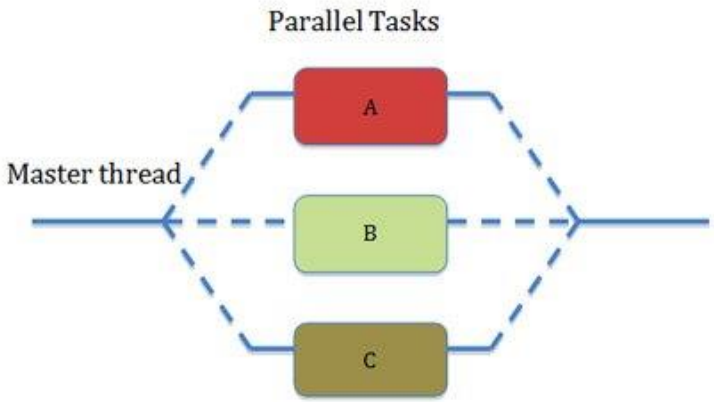
Primer 1 – "Zdravo OpenMP svete!"

```
#include <iostream>
using namespace std;

int main() {

		#pragma omp parallel		num_threads(8)
		{
			cout << "Zdravo OpenMP svete!\n";
		}
		return 0;
}
```

- Broj niti standardno odgovara broju jezgara CPU, zavisi od implementacije
- Model rada **račvanje-spajanje**
- **Master nit** i **paralelna regija**
- Na kraju paralelne regije, sinhronizacija i terminacija svih niti osim master niti



Primer 2 – Rad sa nitima

- Program koji prikazuje identifikatore niti i ukupan broj niti:

```
#include <iostream>
#include <omp.h>
using namespace std;

int main() {
		#pragma omp parallel num_threads(8)
		{
			int id = omp_get_thread_num();
			int podatak = id;
			int brojNiti = omp_get_num_threads();
		cout << "Pozdrav od niti broj " + to_string(id) + ” od ukupno ” + to_string(brojNiti) +” niti.” +
			”. Radim nad podatkom " + to_string(podatak) +
		“.\n”;
		}
		cout << "Kraj paralelne regije!" << endl;
		return 0;
}
```

Primer 2 – Rad sa nitima

- Program koji prikazuje identifikatore niti i ukupan broj niti:

```
#include <iostream>
#include <omp.h>
using namespace std;

int main() {
		int podatak;
		#pragma omp parallel num_threads(8)
		{
			int id = omp_get_thread_num();
			podatak = id;
			int brojNiti = omp_get_num_threads();
		cout << "Pozdrav od niti broj " + to_string(id) + ” od ukupno ” + to_string(brojNiti) +” niti.” +
			”. Radim nad podatkom " + to_string(podatak) +
		“.\n”;
		}
		cout << "Kraj paralelne regije!" << endl;
		return 0;
}
```

Primer 2 – Rad sa nitima

- Rešenje korišćenjem klauzule `private`:

```
#include <iostream>
#include <omp.h>
using namespace std;

int main() {
		int id, podatak, brojNiti;
		#pragma omp parallel private(id, podatak, brojNiti) num_threads(8)
		{
			id = omp_get_thread_num();
			podatak = id;
			brojNiti = omp_get_num_threads();
		cout <<	"Pozdrav od niti broj " + to_string(id) + ” od 		ukupno ”
+ to_string(brojNiti) +” niti.” +
					”. Radim nad podatkom " + to_string(podatak) +
		“.\n”;
		}
		cout << "Kraj paralelne regije!" << endl;
		return 0;
}
```

Primer 2 – Rad sa nitima

- Rešenje korišćenjem direktive `omp critical`:

```
#include <iostream>
#include <omp.h>
using namespace std;

int main() {
		int podatak;
		#pragma omp parallel num_threads(8)
		{
				int id = omp_get_thread_num();
				int brojNiti = omp_get_num_threads();
				#pragma omp critical
				{
						podatak = id;
						cout <<		"Pozdrav od niti broj " + to_string(id) + ” od
ukupno ” + to_string(brojNiti) +” niti.” +
						”. Radim nad podatkom " + to_string(podatak) + “.\n”;
				}
		}
		cout << "Kraj paralelne regije!" << endl;
		return 0;
}
```

Paralelno programiranje sa OpenMP – rezime

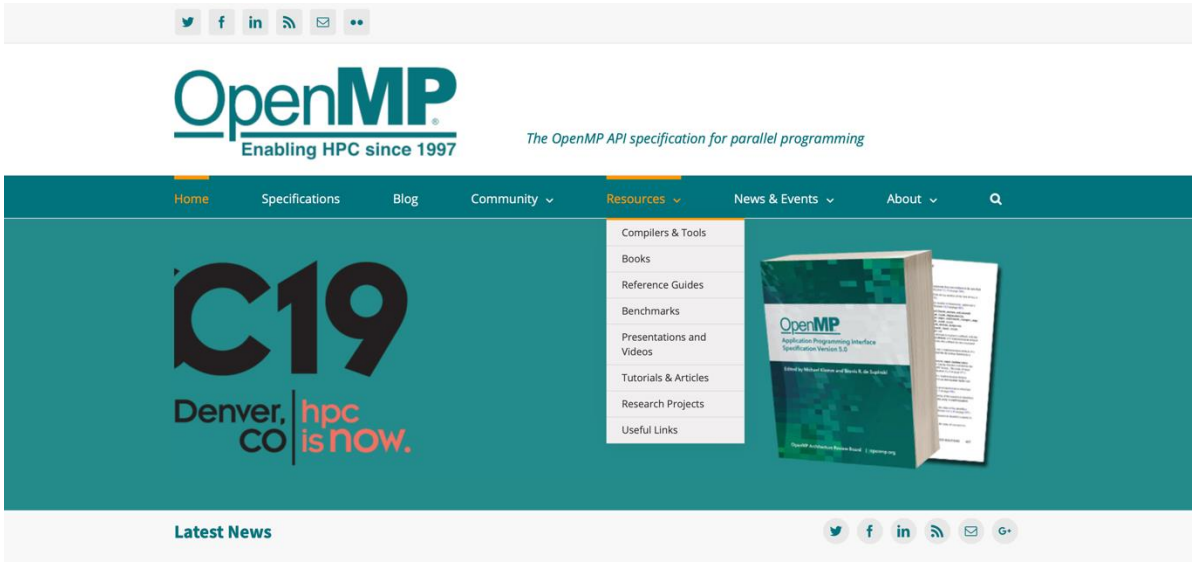
Prednosti i mane

- **Prednosti:**
  - **prenosivi višenitni kod**
  - **direktive automatski** upravljaju **dekompozicijom i raspodelom podataka**
  - **inkrementalni paralelizam** (može se uvoditi po delovima koda)
  - **jedinstven kod za serijske i paralelne programe**
  - kod može da se **izvršava na akceleratorima** (GPU, FPGA, DSP,…)
- **Mane:**
  - **isključivo** za **paralelne sisteme sa deljenom memorijom**
  - neophodan **prevodilac** sa **podrškom za OpenMP**
  - **rizik za uvođenje grešaka** koje se **teško debagiraju** (problemi sinhronizacije i uslova trke)

Izvor: https://en.wikipedia.org/wiki/OpenMP

Resursi

- Kompajleri: https://www.openmp.org/resources/openmp-compilers-tools/
- Knjige: https://www.openmp.org/resources/openmp-books/
- Prezentacije i video materijali: https://www.openmp.org/resources/openmp-presentations/
- Tutorijali: https://www.openmp.org/resources/tutorials-articles/

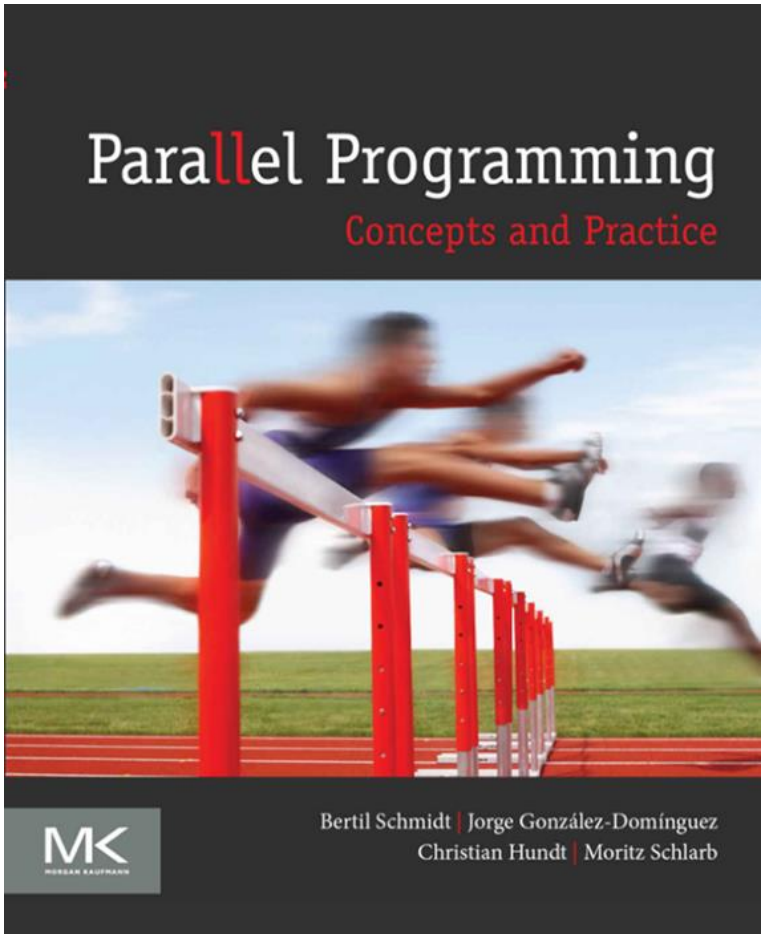


Online kurs

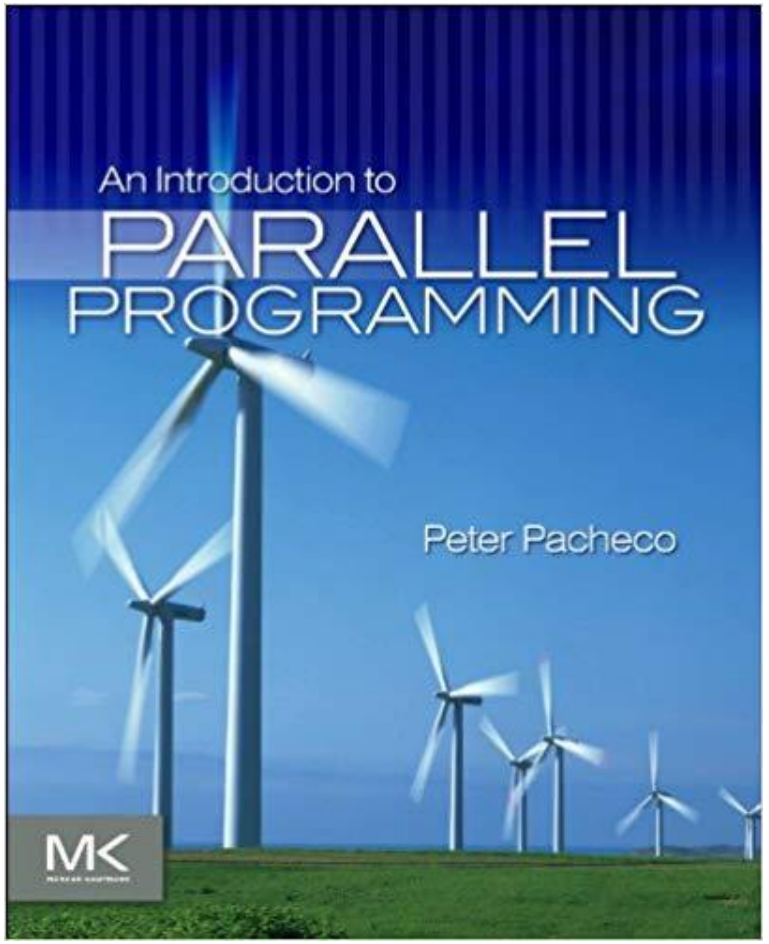
- Tim Mattson (Intel), jedan od kreatora OpenMP, kurs **"Introduction to OpenMP"**:

  https://www.youtube.com/playlist?list=PLLX-Q6B8xqZ8n8bwjGdzBJ25X2utwnoEG

Literatura



https://parallelprogrammingbook.org/



https://www.cs.usfca.edu/~peter/ipp/