Konstrukcija kompilatora

— Optimizacija koda —

Milena Vujošević Janičić

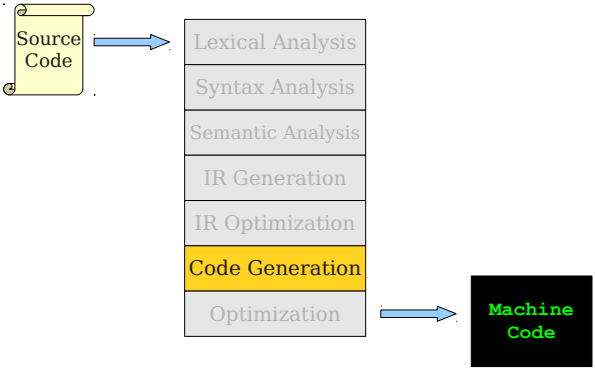
www.matf.bg.ac.rs/~milena

Matematički fakultet, Univerzitet u Beogradu

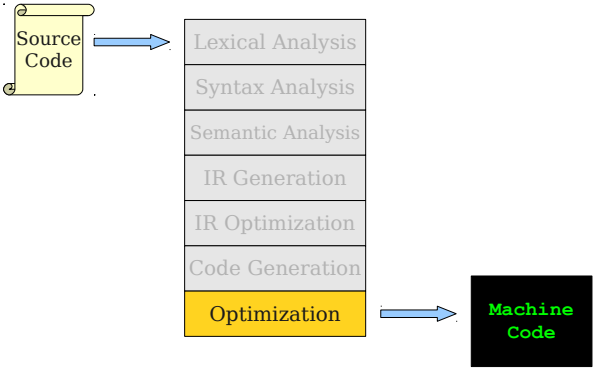
Pregled

Pregled

Where We Are


Source Code
Lexical Analysis
Syntax Analysis
Semantic Analysis
IR Generation
IR Optimization
Code Generation
Optimization
Machine Code

Where We Are


Source Code
Lexical Analysis
Syntax Analysis
Semantic Analysis
IR Generation
IR Optimization
Code Generation
Optimization
Machine Code

Finalna optimizacija koda

- Cilj: Optimizovati generisani kod korišćenjem mašinski zavisnih osobina koje nisu vidljive na IR nivou
- Kritičan korak većine kompajlera, ali obično veoma neuredan:
  - Tehnike koje se razviju za jednu mašinu mogu da budu potpuni nekorisne za drugu
  - Tehnike koje se razviju za jedan jezik mogu da budu potpuno nekorisne za drugi
- Razmotrićemo optimizaciju raspoređivanja instrukcija
- Postoje i razne druge optimizacije koda, npr optimizacije sa ciljem boljeg upravljanja kešom

Pregled

Processor Pipelines

Processor Pipelines

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t5, $t3, $t4   # $t5 = $t3 + $t4
add $t8, $t6, $t7   # $t8 = $t6 + $t7
```

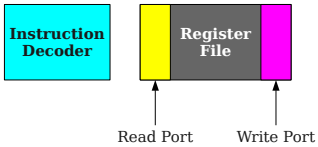
Processor Pipelines


```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t5, $t3, $t4   # $t5 = $t3 + $t4
add $t8, $t6, $t7   # $t8 = $t6 + $t7
```

Processor Pipelines

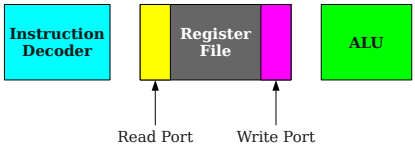

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t5, $t3, $t4   # $t5 = $t3 + $t4
add $t8, $t6, $t7   # $t8 = $t6 + $t7
```

Processor Pipelines



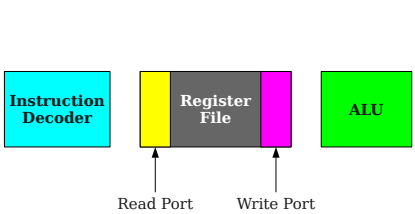

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t5, $t3, $t4   # $t5 = $t3 + $t4
add $t8, $t6, $t7   # $t8 = $t6 + $t7
```

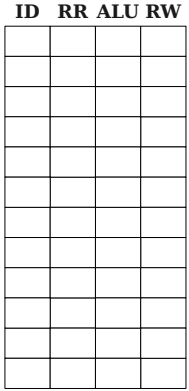
Processor Pipelines




```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t5, $t3, $t4   # $t5 = $t3 + $t4
add $t8, $t6, $t7   # $t8 = $t6 + $t7
```

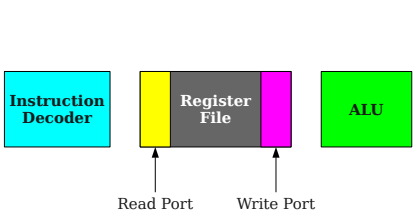
Processor Pipelines

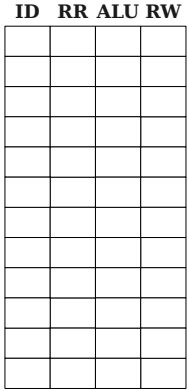





```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t5, $t3, $t4   # $t5 = $t3 + $t4
add $t8, $t6, $t7   # $t8 = $t6 + $t7
```

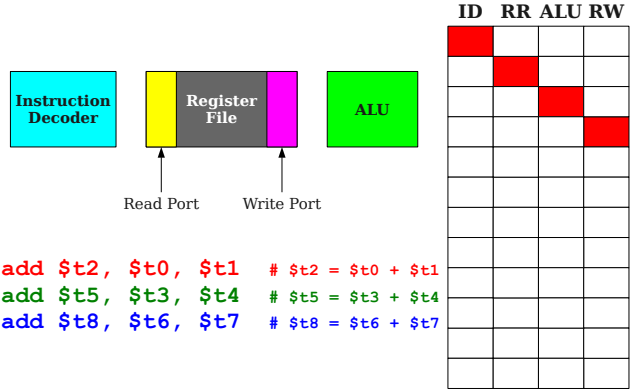
Processor Pipelines





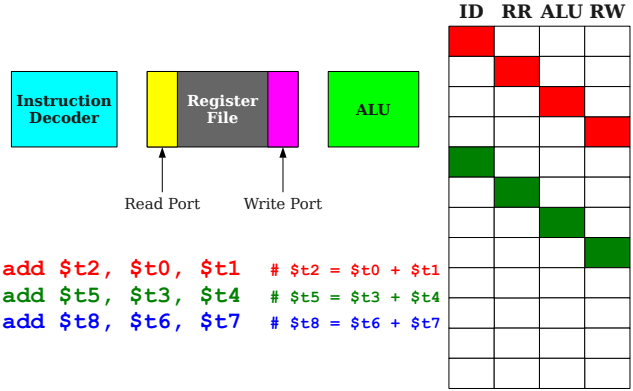

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t5, $t3, $t4   # $t5 = $t3 + $t4
add $t8, $t6, $t7   # $t8 = $t6 + $t7
```

Processor Pipelines



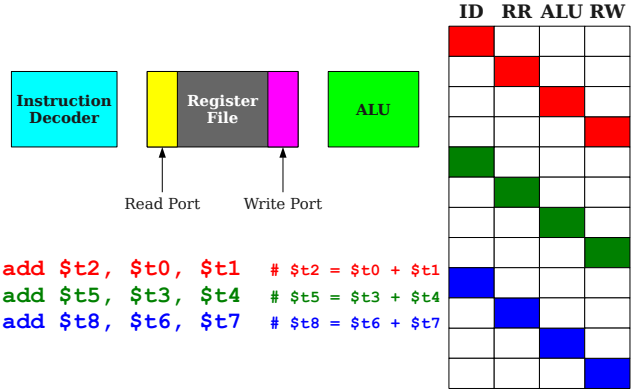

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t5, $t3, $t4   # $t5 = $t3 + $t4
add $t8, $t6, $t7   # $t8 = $t6 + $t7
```

Processor Pipelines



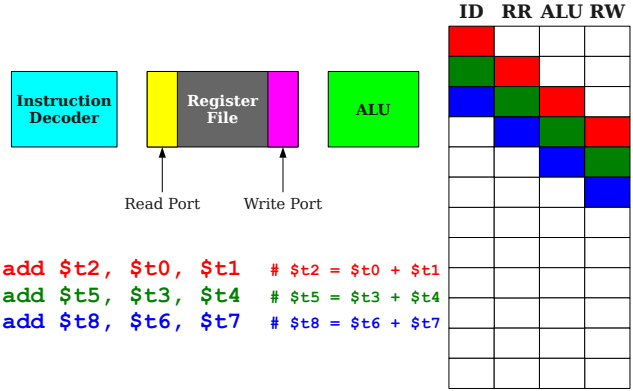

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t5, $t3, $t4   # $t5 = $t3 + $t4
add $t8, $t6, $t7   # $t8 = $t6 + $t7
```

Processor Pipelines




```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t5, $t3, $t4   # $t5 = $t3 + $t4
add $t8, $t6, $t7   # $t8 = $t6 + $t7
```

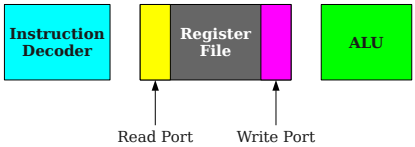
Processor Pipelines



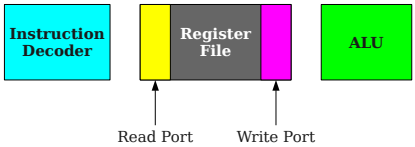

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t5, $t3, $t4   # $t5 = $t3 + $t4
add $t8, $t6, $t7   # $t8 = $t6 + $t7
```

Pipeline Hazards

Pipeline Hazards

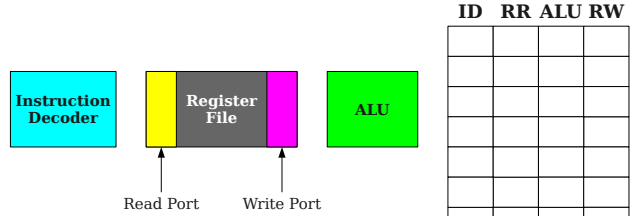


Pipeline Hazards



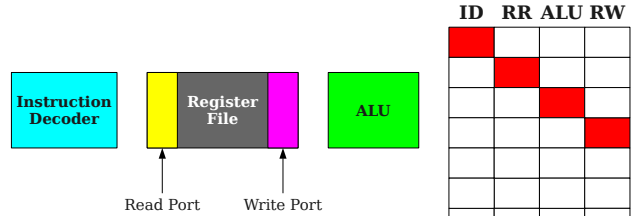

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



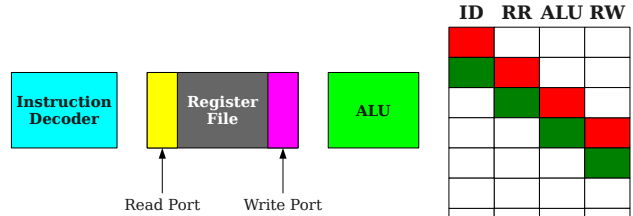

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



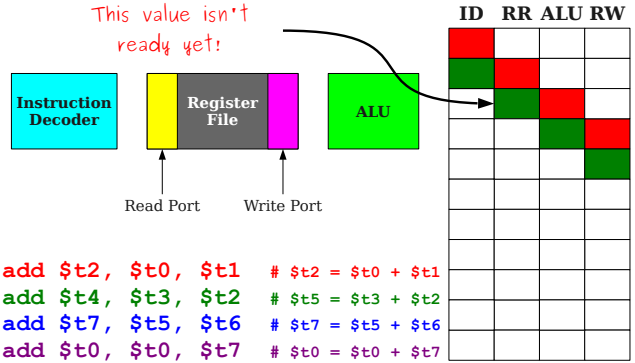

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



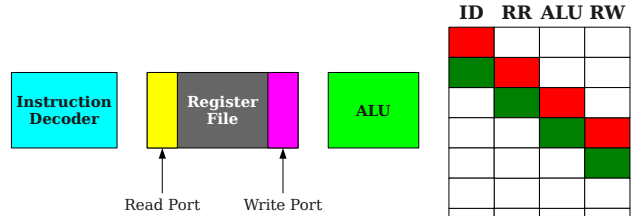

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



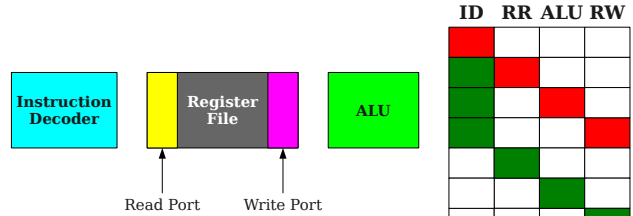

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



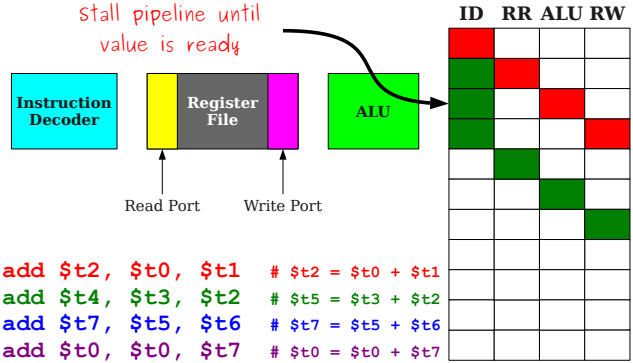

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



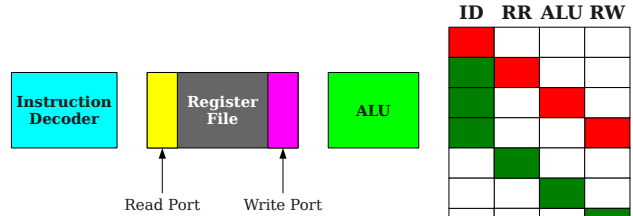

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



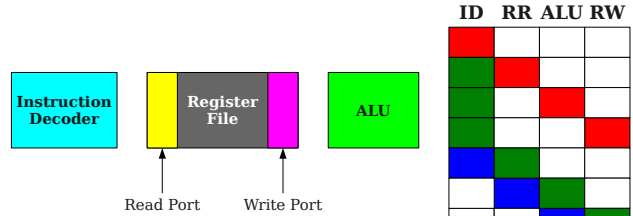

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



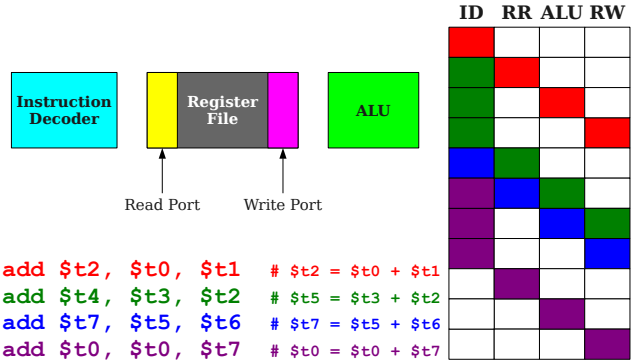

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



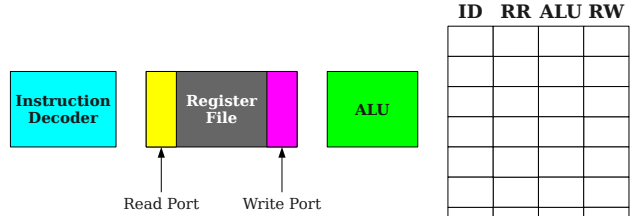

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



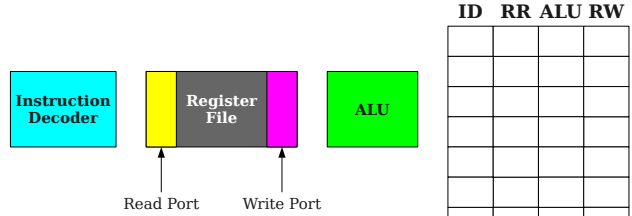

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



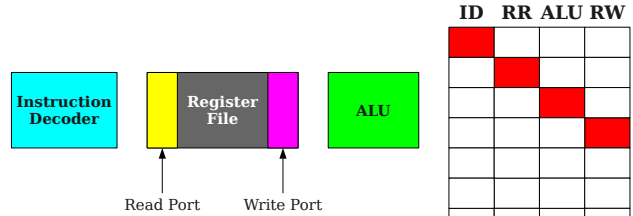

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



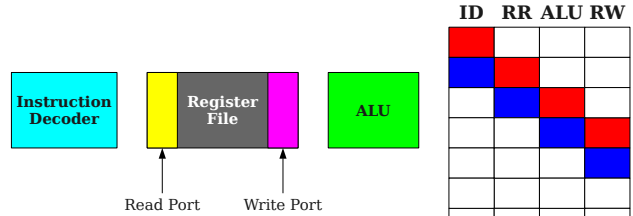

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



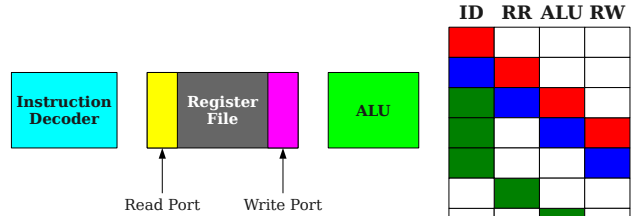

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



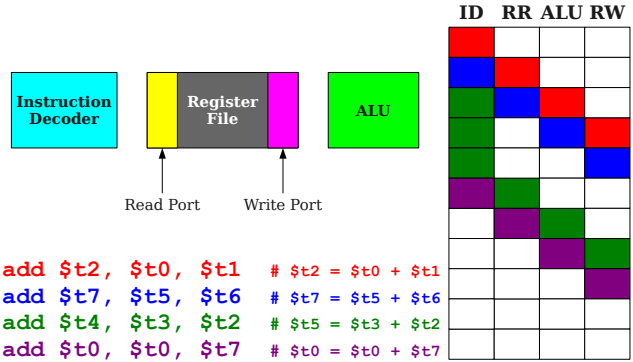

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



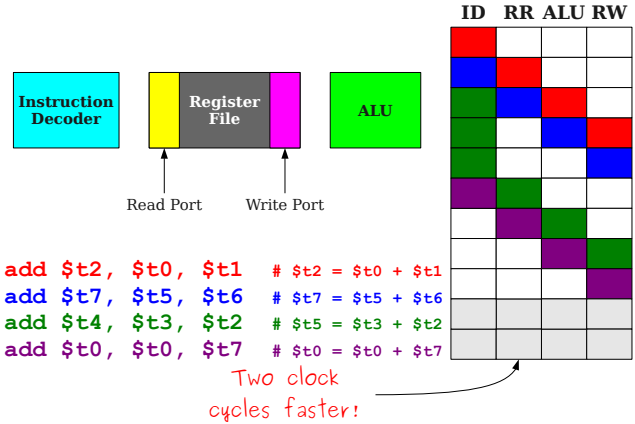

```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards




```
add $t2, $t0, $t1   # $t2 = $t0 + $t1
add $t7, $t5, $t6   # $t7 = $t5 + $t6
add $t4, $t3, $t2   # $t5 = $t3 + $t2
add $t0, $t0, $t7   # $t0 = $t0 + $t7
```

Pipeline Hazards



Raspoređivanje instrukcija

- Zbog procesorskog pipilining-a, redosled u kojem se instrukcije izvršavaju može da utiče na performanse
- **Raspoređivanje instrukcija je pravljenje rasporeda instrukcija sa ciljem da se poboljšaju performanse**
- Svi dobri kompajleri imaju neku vrstu podrške za raspoređivanje instrukcija

Zavisnost među podacima

- Zavisnost među podacima u mašinskom kodu je skup instrukcija čije ponašanje zavisi jedno od druge
- Intuitivno, skup instrukcija koje ne mogu da se poređaju na drugačiji način
- Postoje tri vrste zavisnosti: čitanje nakon pisanja, pisanje nakon čitanja, pisanje nakon pisanja

Finding Data Dependencies

```
t0 = t1 + t2
t1 = t0 + t1
t3 = t2 + t4
t0 = t1 + t2
t5 = t3 + t4
t6 = t2 + t3
```

Finding Data Dependencies

```
t0 = t1 + t2

t1 = t0 + t1

t3 = t2 + t4

t0 = t1 + t2

t5 = t3 + t4

t6 = t2 + t7
```

Finding Data Dependencies

```
t0 = t1 + t2
```

```
t1 = t0 + t1
```

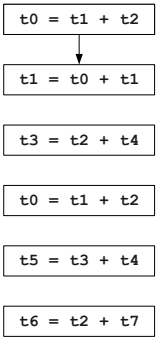
```
t3 = t2 + t4
```

```
t0 = t1 + t2
```

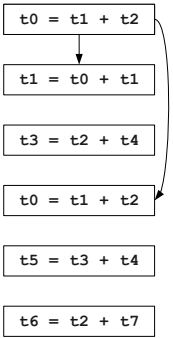
```
t5 = t3 + t4
```

```
t6 = t2 + t7
```

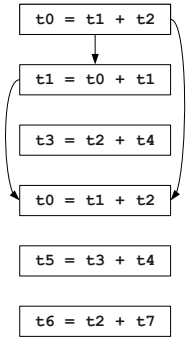
Finding Data Dependencies



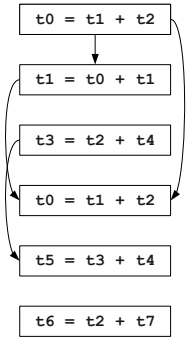
Finding Data Dependencies



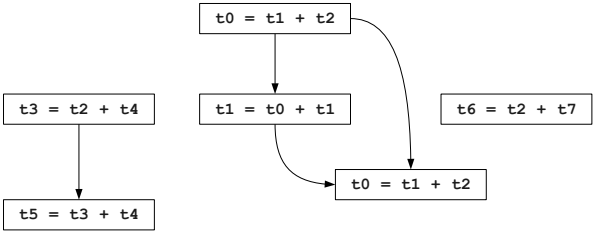
Finding Data Dependencies



Finding Data Dependencies



Finding Data Dependencies

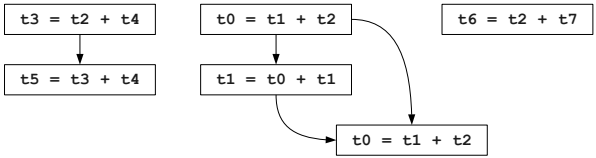


Graf zavisnosti podataka

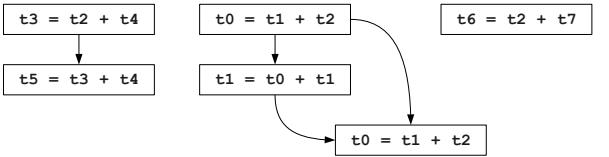
- Graf koji prikazuje zavisnosti podataka u okviru osnovnog bloka naziva se graf zavisnosti podataka
- To je direktan aciklični graf. Direktan, jer uvek jedna instrukcija zavisi od druge. Acikličan, jer nisu moguće ciklične zavisnosti.
- Mogu se rasporediti instrukcije u okviru osnovnog bloka u bilo kom redosledu sve dok se ne raspodele instrukcije tako da neka instrukcija prethodi svom roditelju
- Ideja: **napravi topološko sortiranje zavisnosti podataka i poređaj instrukcije u tom redosledu**

Instruction Scheduling

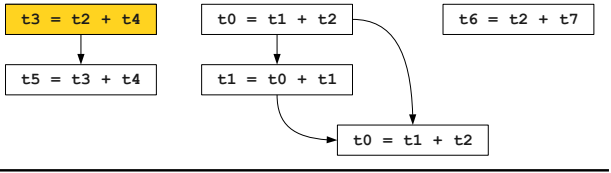
Instruction Scheduling



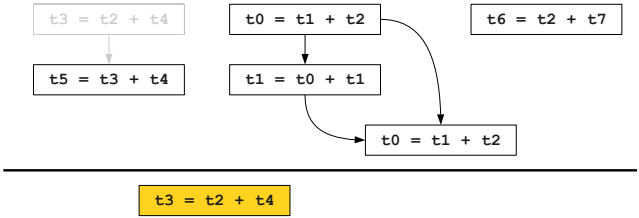
Instruction Scheduling



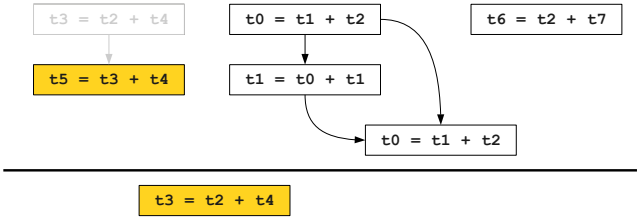
Instruction Scheduling



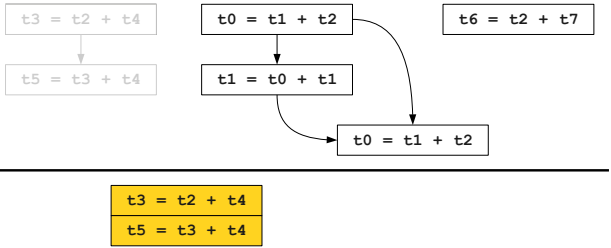
Instruction Scheduling



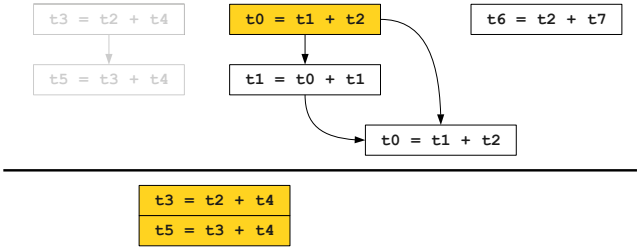
Instruction Scheduling



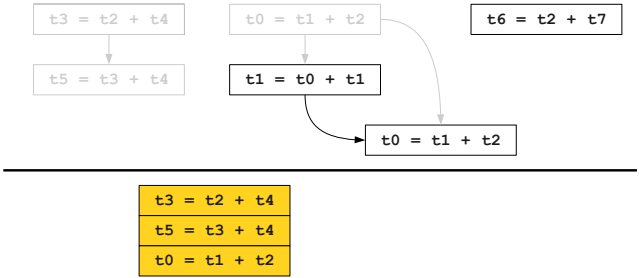
Instruction Scheduling



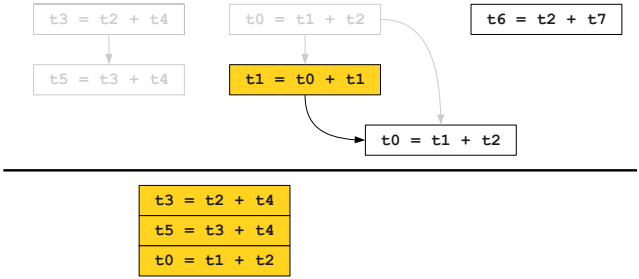
Instruction Scheduling



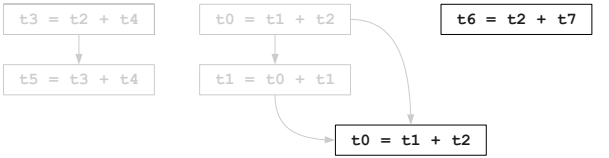
Instruction Scheduling



Instruction Scheduling

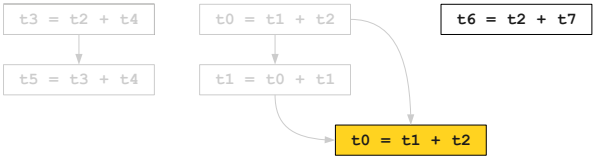


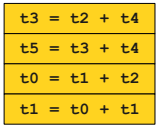
Instruction Scheduling



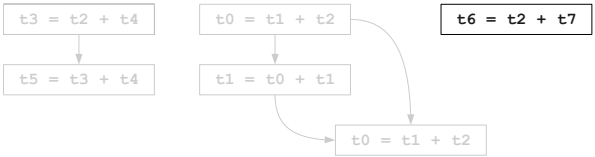
t3 = t2 + t4
t5 = t3 + t4
t0 = t1 + t2
t1 = t0 + t1

Instruction Scheduling



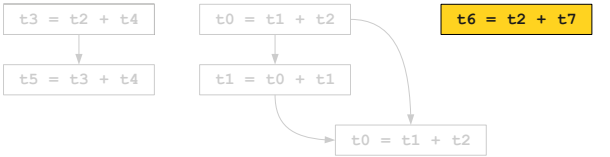


Instruction Scheduling



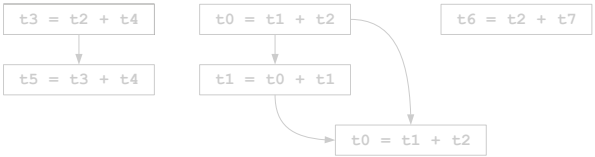

```
t3 = t2 + t4
t5 = t3 + t4
t0 = t1 + t2
t1 = t0 + t1
t0 = t1 + t2
```

Instruction Scheduling



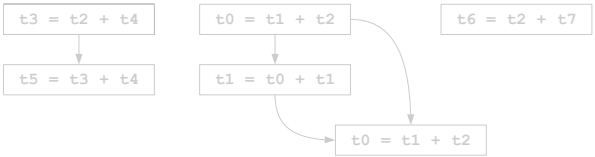

```
t3 = t2 + t4
t5 = t3 + t4
t0 = t1 + t2
t1 = t0 + t1
t0 = t1 + t2
```

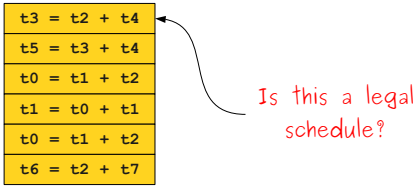
Instruction Scheduling



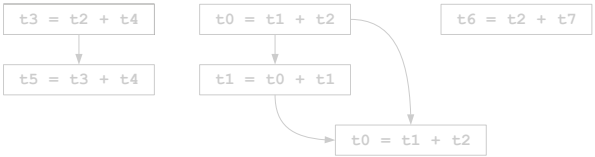

```
t3 = t2 + t4
t5 = t3 + t4
t0 = t1 + t2
t1 = t0 + t1
t0 = t1 + t2
t6 = t2 + t7
```

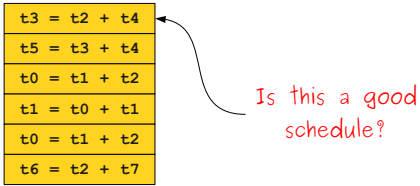
Instruction Scheduling



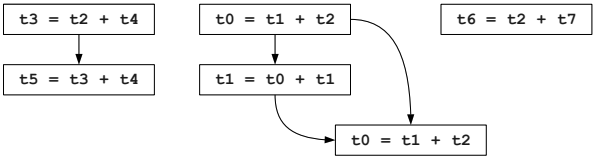


Instruction Scheduling



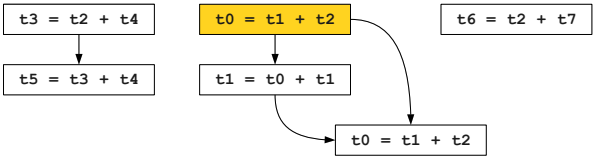


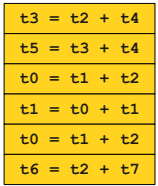
Instruction Scheduling



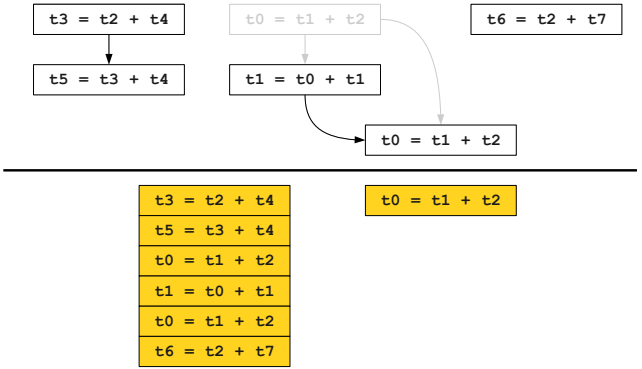

```
t3 = t2 + t4
t5 = t3 + t4
t0 = t1 + t2
t1 = t0 + t1
t0 = t1 + t2
t6 = t2 + t7
```

Instruction Scheduling

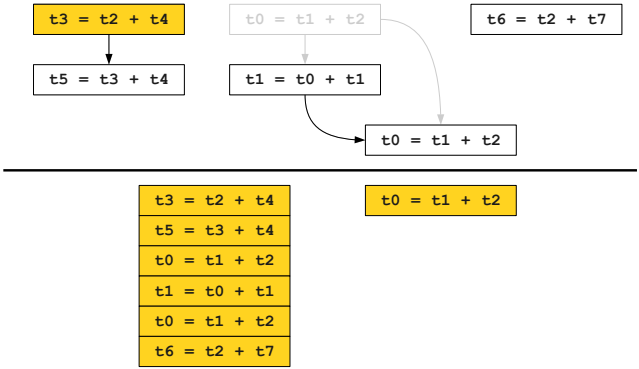




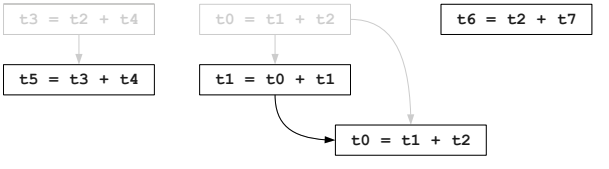
Instruction Scheduling



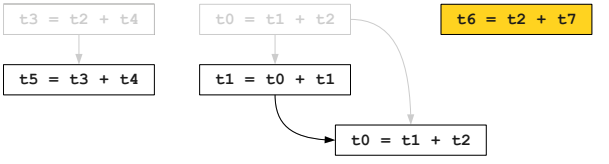
Instruction Scheduling

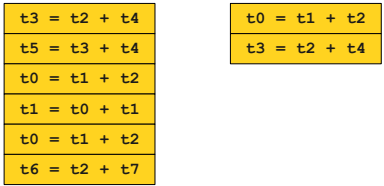


Instruction Scheduling

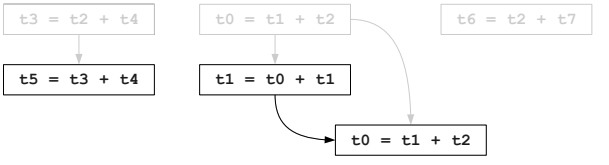


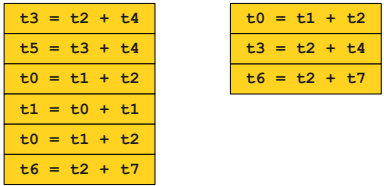
Instruction Scheduling



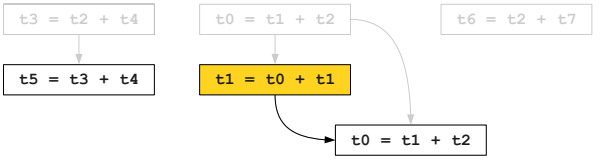


Instruction Scheduling

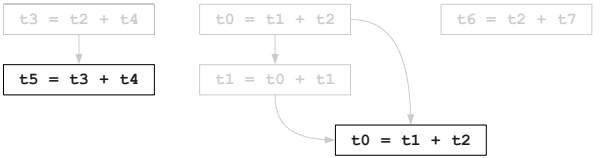


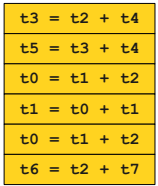


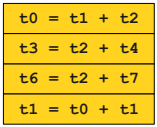
Instruction Scheduling



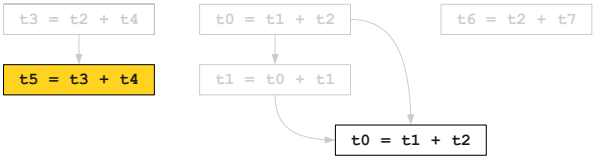
Instruction Scheduling

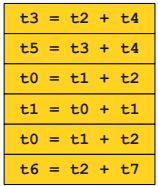


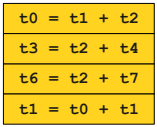




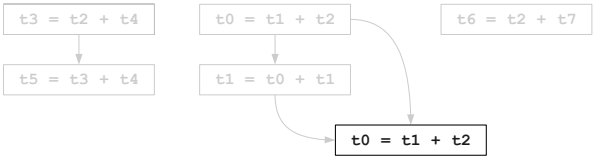
Instruction Scheduling







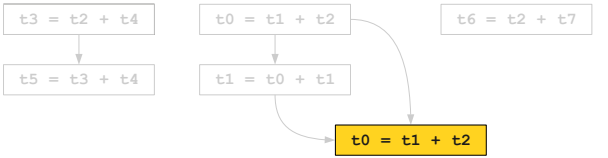
Instruction Scheduling




```
t3 = t2 + t4
t5 = t3 + t4
t0 = t1 + t2
t1 = t0 + t1
t0 = t1 + t2
t6 = t2 + t7
```

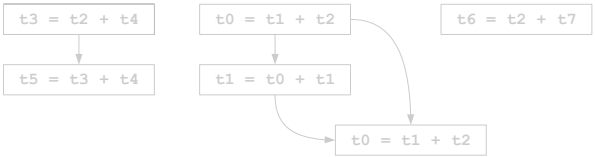
```
t0 = t1 + t2
t3 = t2 + t4
t6 = t2 + t7
t1 = t0 + t1
t5 = t3 + t4
```

Instruction Scheduling



t3 = t2 + t4	t0 = t1 + t2
t5 = t3 + t4	t3 = t2 + t4
t0 = t1 + t2	t6 = t2 + t7
t1 = t0 + t1	t1 = t0 + t1
t0 = t1 + t2	t5 = t3 + t4
t6 = t2 + t7	

Instruction Scheduling



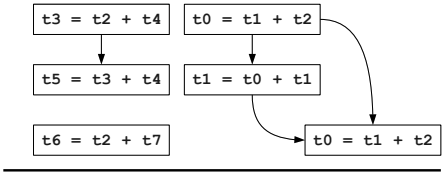
t3 = t2 + t4
t5 = t3 + t4
t0 = t1 + t2
t1 = t0 + t1
t0 = t1 + t2
t6 = t2 + t7

t0 = t1 + t2
t3 = t2 + t4
t6 = t2 + t7
t1 = t0 + t1
t5 = t3 + t4
t0 = t1 + t2

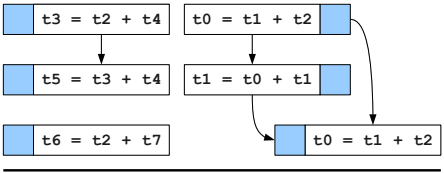
Problem

- Može postojati puno ispravnih topoloških uređenja grafa zavisnosti podataka
- Kako izabrati onaj redosled koji je dobar?
- U opštem slučaju, pronalaženje najboljeg rasporeda instrukcija je NP-težak problem.
- U praksi se koriste heuristike

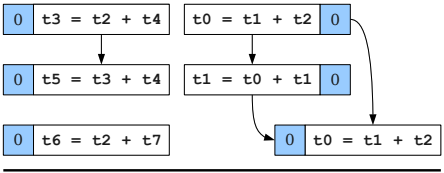
Instruction Scheduling



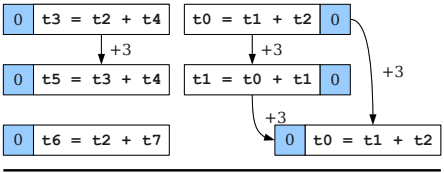
Instruction Scheduling



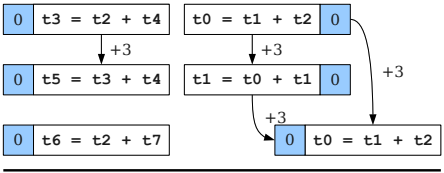
Instruction Scheduling

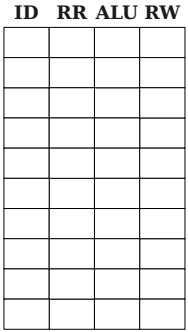


Instruction Scheduling

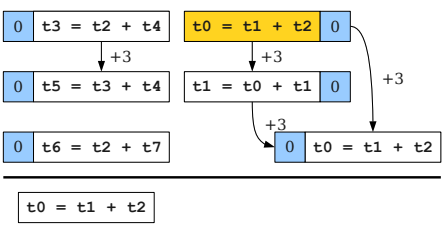


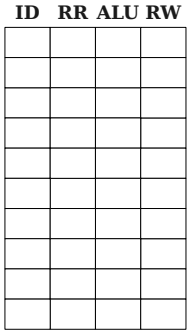
Instruction Scheduling



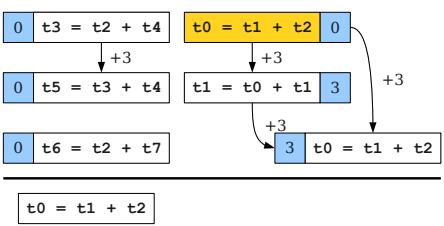


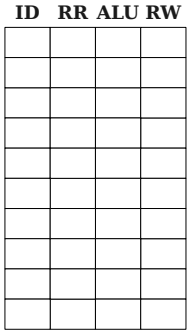
Instruction Scheduling



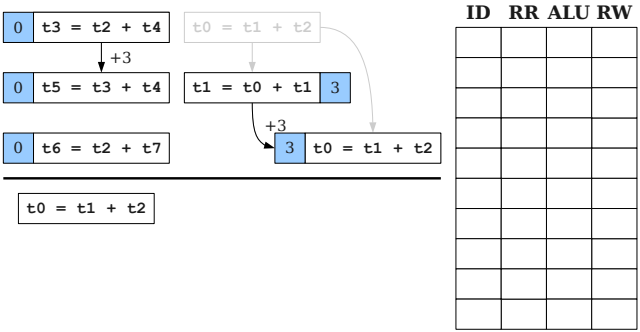


Instruction Scheduling

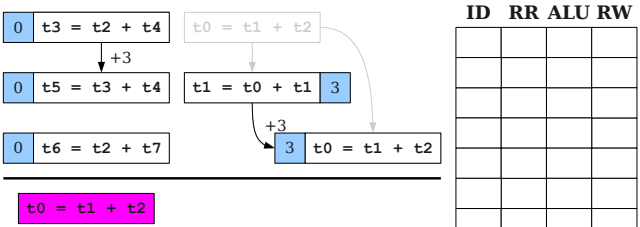




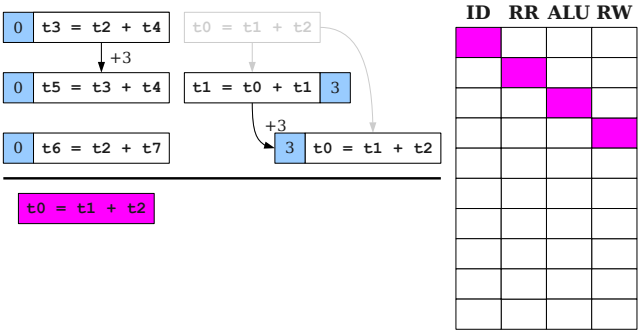
Instruction Scheduling



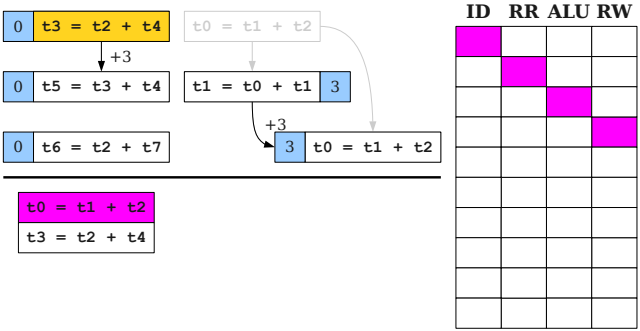
Instruction Scheduling



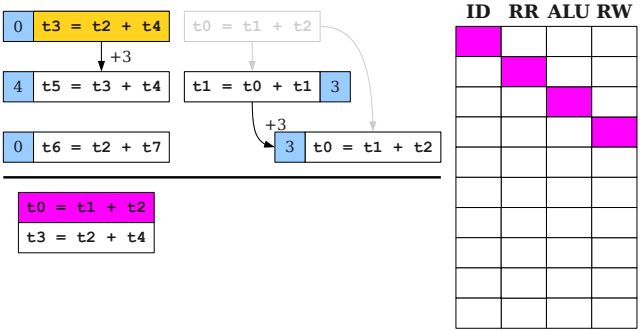
Instruction Scheduling



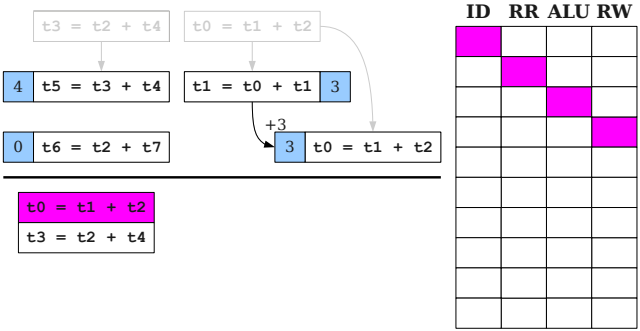
Instruction Scheduling



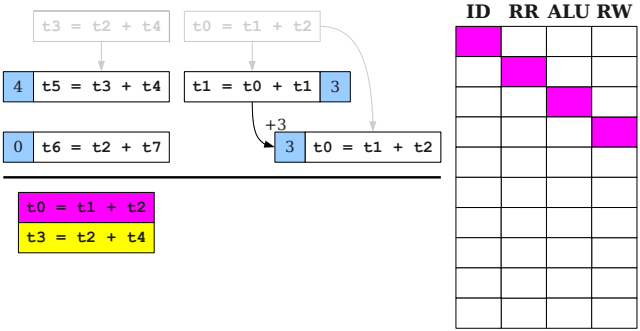
Instruction Scheduling



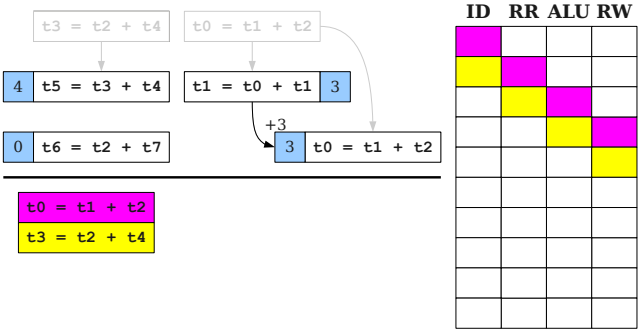
Instruction Scheduling



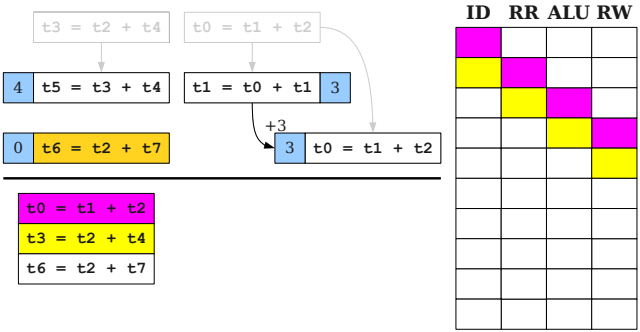
Instruction Scheduling



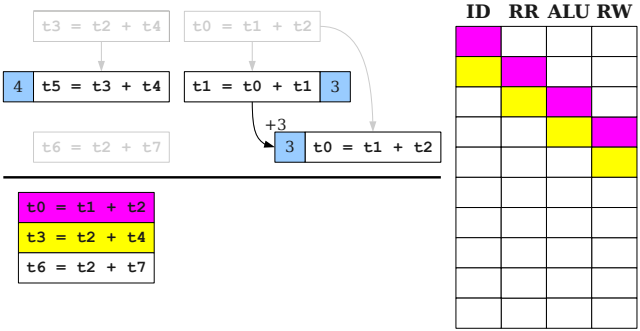
Instruction Scheduling



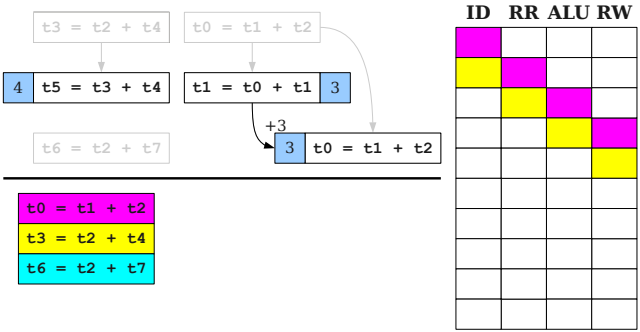
Instruction Scheduling



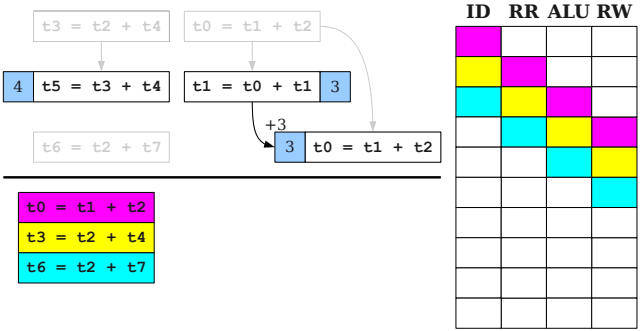
Instruction Scheduling



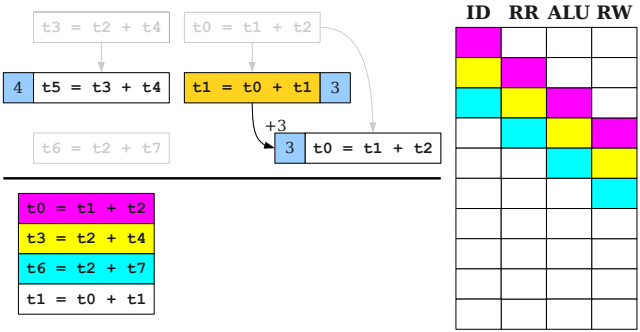
Instruction Scheduling



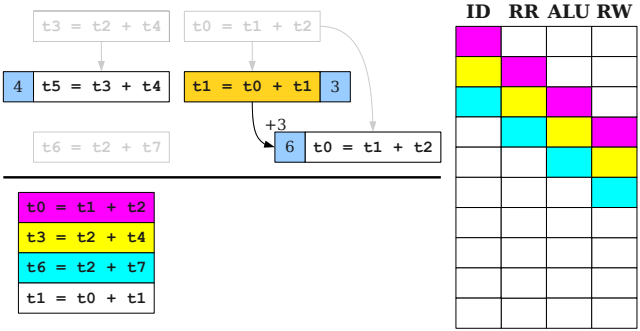
Instruction Scheduling



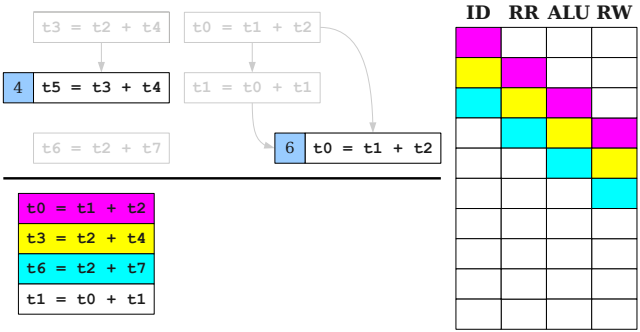
Instruction Scheduling



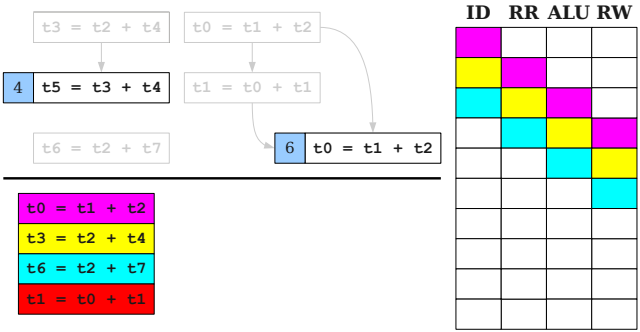
Instruction Scheduling



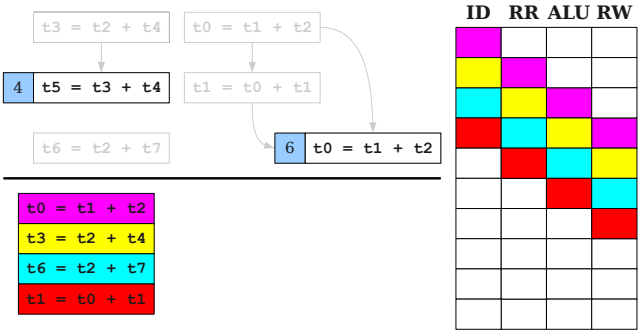
Instruction Scheduling



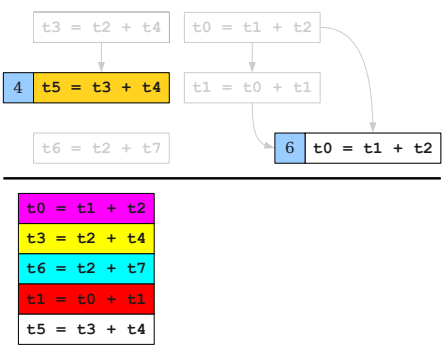
Instruction Scheduling



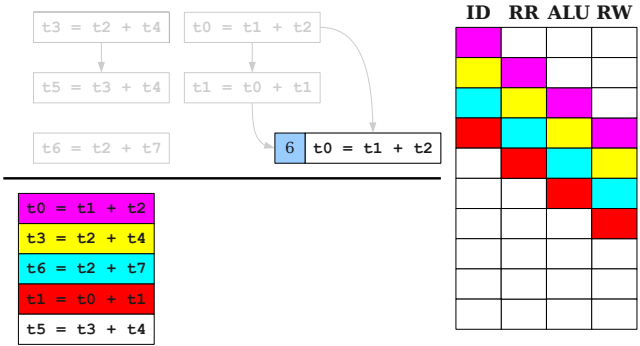
Instruction Scheduling



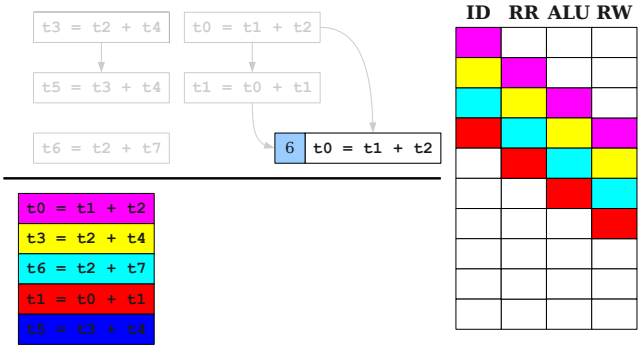
Instruction Scheduling



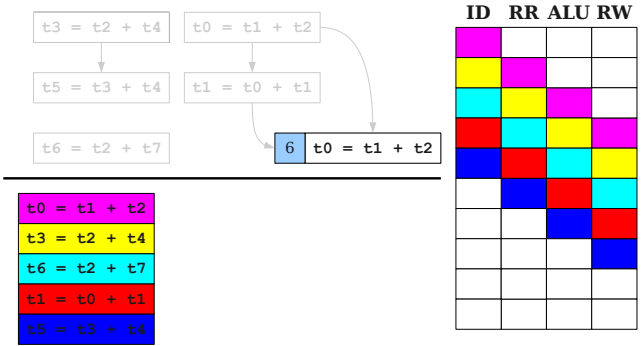
Instruction Scheduling



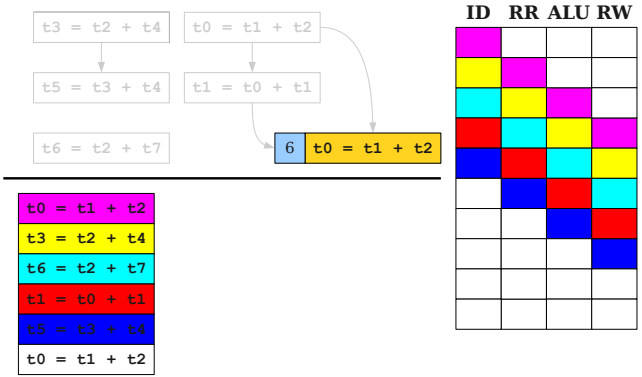
Instruction Scheduling



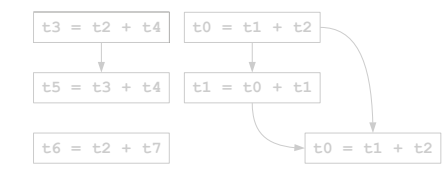
Instruction Scheduling

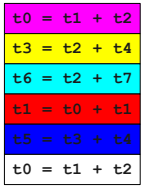


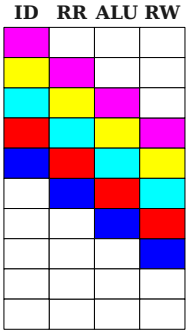
Instruction Scheduling



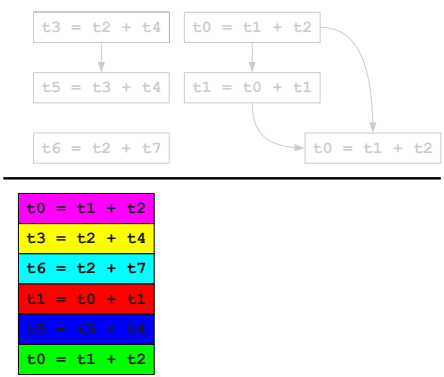
Instruction Scheduling



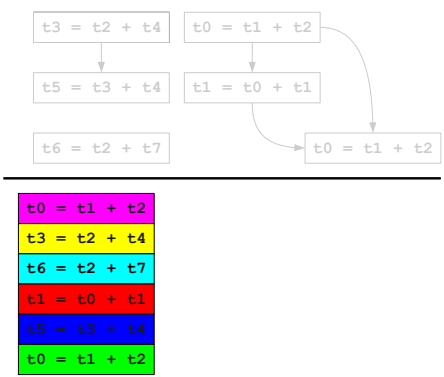




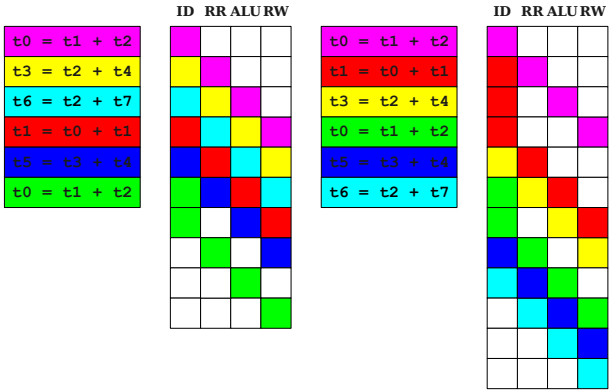
Instruction Scheduling



Instruction Scheduling



For Comparison



Raspoređivanje instrukcija

- Moderni kompajleri mogu da rade i značajno agresivnije raspoređivanje instrukcija sa ciljem da se dobiju bolje performanse programa
- Primeri ovih tehnika su razmotavanje petlji i software pipelining

Pregled

Upotreba keša

- Pored optimizacije raspoređivanje instrukcija, mogu se vršiti i optimizacije transformacije koda sa ciljem bolje upotrebe keša
- Upotreba keša se zasniva na dve vrste lokalnosti: vremenska i prostorna. Vremenska: ako je nekoj memoriji skoro pristupano, verovatno će joj biti ponovo pristupano uskoro. Prostorna: ako je nekoj memoriji skoro pristupano, verovatno će i njeni susedni objekti biti takođe uskoro korišćeni.
- Većina keš memorija je dizajnirano da iskoristi ove lokalnosti tako što se u kešu drže skoro adresirani objekti i tako što se u keš ubacuje i sadržaj memorije u blizini

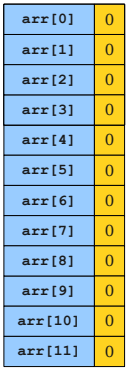
Memory Caches

Memory Caches

```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```

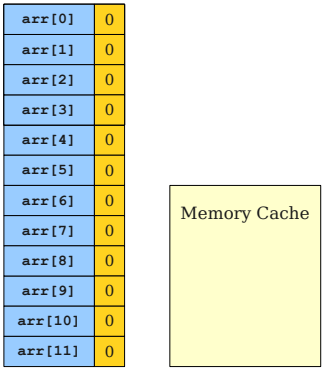
Memory Caches

```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```



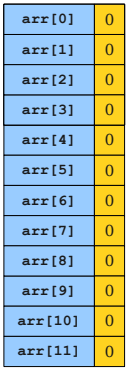
Memory Caches

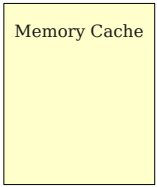
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```



Memory Caches

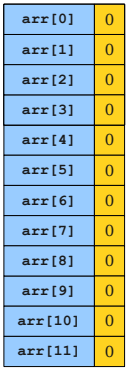
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```

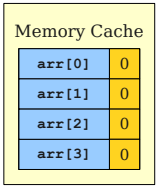




Memory Caches

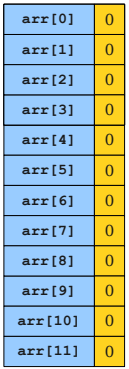
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```

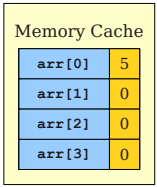




Memory Caches

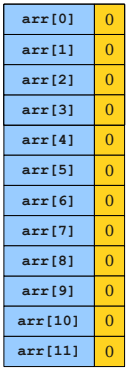
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```

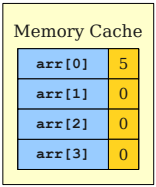




Memory Caches

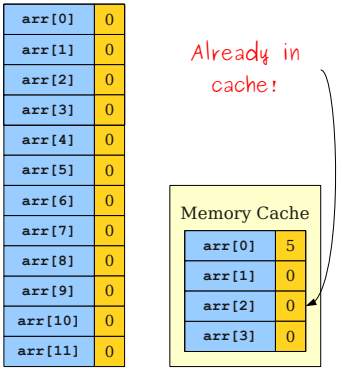
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```





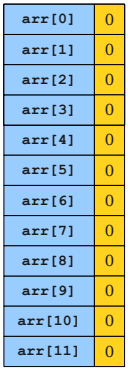
Memory Caches

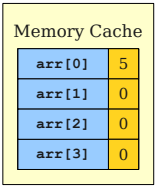
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```



Memory Caches

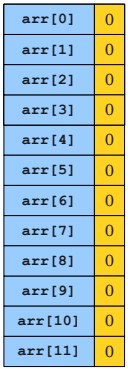
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```

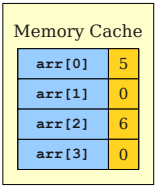




Memory Caches

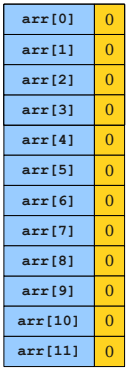
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```

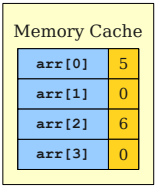




Memory Caches

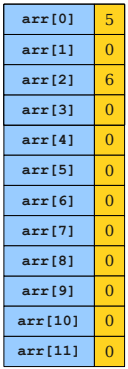
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```





Memory Caches

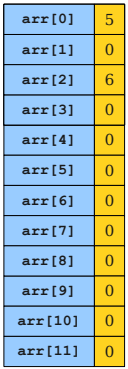
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```

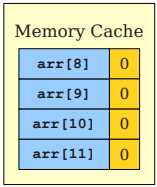


Memory Cache

Memory Caches

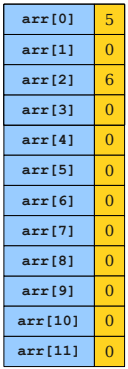
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```

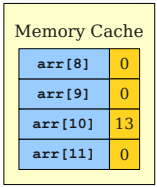




Memory Caches

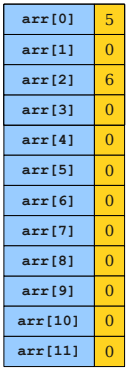
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```

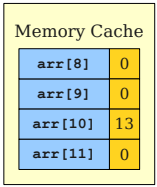




Memory Caches

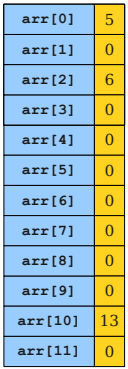
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```





Memory Caches

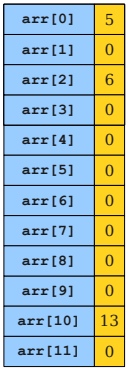
```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```

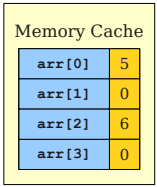


Memory Cache

Memory Caches

```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```





Memory Caches

```
arr[0]  = 5;
arr[2]  = 6;
arr[10] = 13;
arr[1]  = 4;
```

arr[0]	5
arr[1]	0
arr[2]	6
arr[3]	0
arr[4]	0
arr[5]	0
arr[6]	0
arr[7]	0
arr[8]	0
arr[9]	0
arr[10]	13
arr[11]	0

Memory Cache

arr[0]	5
arr[1]	4
arr[2]	6
arr[3]	0

The Problem with Caches

The Problem with Caches

```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```

The Problem with Caches

```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```

00	01	02	03
10	11	12	13
20	21	22	23
30	31	32	33

The Problem with Caches

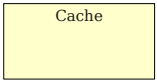
```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```

00	01	02	03	10	11	12	13	20	21	22	23	30	31	32	33

The Problem with Caches

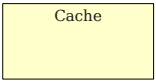
```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```

00	01	02	03	10	11	12	13	20	21	22	23	30	31	32	33



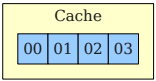
The Problem with Caches

```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```



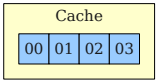
The Problem with Caches

```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```



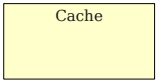
The Problem with Caches

```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```



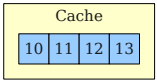
The Problem with Caches

```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```



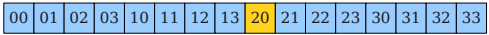
The Problem with Caches

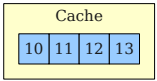
```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```



The Problem with Caches

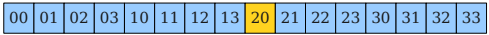
```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```

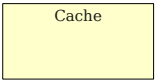




The Problem with Caches

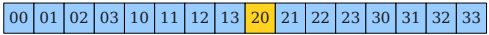
```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```

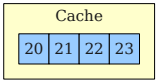




The Problem with Caches

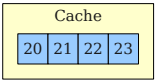
```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```





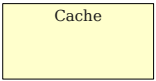
The Problem with Caches

```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```



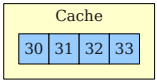
The Problem with Caches

```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```



The Problem with Caches

```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```



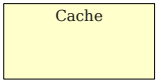
Poboljšanje lokalnosti

- Programeri obično pišu kod bez razumevanja o posledicama lokalnosti jer jezici ne prikazuju detalje memorije
- Neki kompajleri su sposobni da prepišu kod tako da se lokalnost iskoristi
- Primer takve optimizacije je preraspoređivanje petlji

Loop Reordering

```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```

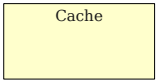
00	01	02	03	10	11	12	13	20	21	22	23	30	31	32	33



Loop Reordering

```
int[][] array;
for (j = 0; j < 4; j = j + 1)
    for (i = 0; i < 4; i = i + 1)
        array[i][j] = 0;
```

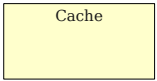
00	01	02	03	10	11	12	13	20	21	22	23	30	31	32	33



Loop Reordering

```
int[][] array;
for (i = 0; i < 4; i = i + 1)
    for (j = 0; j < 4; j = j + 1)
        array[i][j] = 0;
```

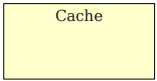
00	01	02	03	10	11	12	13	20	21	22	23	30	31	32	33



Loop Reordering

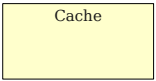
```
int[][] array;
for (i = 0; i < 4; i = i + 1)
    for (j = 0; j < 4; j = j + 1)
        array[i][j] = 0;
```

00	01	02	03	10	11	12	13	20	21	22	23	30	31	32	33



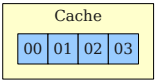
Loop Reordering

```
int[][] array;
for (i = 0; i < 4; i = i + 1)
    for (j = 0; j < 4; j = j + 1)
        array[i][j] = 0;
```



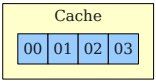
Loop Reordering

```
int[][] array;
for (i = 0; i < 4; i = i + 1)
    for (j = 0; j < 4; j = j + 1)
        array[i][j] = 0;
```



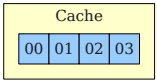
Loop Reordering

```
int[][] array;
for (i = 0; i < 4; i = i + 1)
    for (j = 0; j < 4; j = j + 1)
        array[i][j] = 0;
```



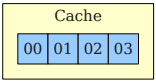
Loop Reordering

```
int[][] array;
for (i = 0; i < 4; i = i + 1)
    for (j = 0; j < 4; j = j + 1)
        array[i][j] = 0;
```



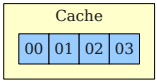
Loop Reordering

```
int[][] array;
for (i = 0; i < 4; i = i + 1)
    for (j = 0; j < 4; j = j + 1)
        array[i][j] = 0;
```



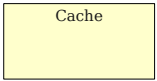
Loop Reordering

```
int[][] array;
for (i = 0; i < 4; i = i + 1)
    for (j = 0; j < 4; j = j + 1)
        array[i][j] = 0;
```



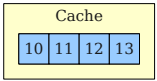
Loop Reordering

```
int[][] array;
for (i = 0; i < 4; i = i + 1)
    for (j = 0; j < 4; j = j + 1)
        array[i][j] = 0;
```



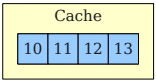
Loop Reordering

```
int[][] array;
for (i = 0; i < 4; i = i + 1)
    for (j = 0; j < 4; j = j + 1)
        array[i][j] = 0;
```



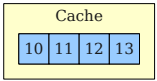
Loop Reordering

```
int[][] array;
for (i = 0; i < 4; i = i + 1)
    for (j = 0; j < 4; j = j + 1)
        array[i][j] = 0;
```



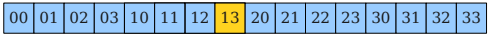
Loop Reordering

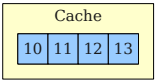
```
int[][] array;
for (i = 0; i < 4; i = i + 1)
    for (j = 0; j < 4; j = j + 1)
        array[i][j] = 0;
```



Loop Reordering

```
int[][] array;
for (i = 0; i < 4; i = i + 1)
    for (j = 0; j < 4; j = j + 1)
        array[i][j] = 0;
```

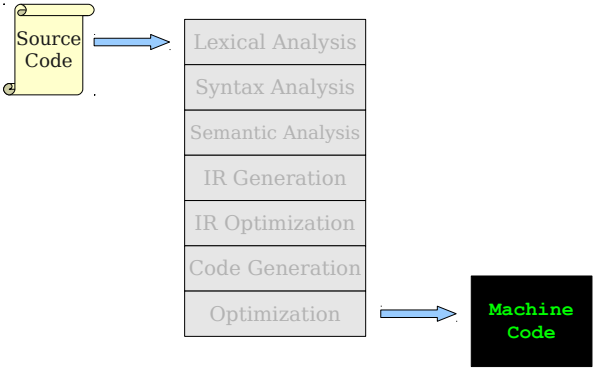




Optimizacije koda

- Postoje i razne druge optimizacije koje se mogu sprovesti na nivou izgenerisanog koda

Where We Are


Source Code
Lexical Analysis
Syntax Analysis
Semantic Analysis
IR Generation
IR Optimization
Code Generation
Optimization
Machine Code

Pregled

Literatura

- (The Dragon Book) Compilers: Principles, Techniques, and Tools Alfred V. Aho,Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman
- Kompajleri Stanford
  `https://web.stanford.edu/class/archive/cs/cs143/cs143.1128/`