

DOMAĆI ZADATAK 2019/2020.

Cilj domaćeg zadatka je formiranje petlje softverske protočnosti za minimalni broj ciklusa. U okviru svake grupe data je *doAll* ili *doAcross* petlja koju treba transformisati u skladu sa zahtevima.

Kao deo rešenja potrebno je prikazati graf petlje nakon primene softverske protočnosti. Transformisana *doAll* petlja treba da se izvršava za jedan ciklus, a *doAcross* za minimalni mogući broj ciklusa. Prikaz rasporeda dati tako da se za operaciju navode svi njeni ciklusi. Raspored napraviti za minimalan nivo softverske protočnosti. Prilikom prikaza rasporeda usvojiti sledeće:

- a. Operacije traju koliko i kašnjenja zavisnosti
- b. Operacije iz originalnih prvih iteracija u predpetlji se označavaju sledećom notacijom Opx i,j , gde je x redni broj operacije u iteraciji, i indeks originalne iteracije, a j ciklus izvršavanja u pipeline-u za tu operaciju.
- c. Operacije iz originalnih prvih iteracija u postpetlji se označavaju sledećom notacijom Opx i,j , gde je x redni broj operacije u iteraciji, i indeks originalne iteracije, a j ciklus izvršavanja u pipeline-u za tu operaciju.
- d. Istu notaciju koristiti i za novu iteraciju, samo indekse iteracije predstaviti pomoću indeksa petlje i , a pritom definisati opseg indeksa petlje.
- e. Pretpostaviti da je u istom ciklusu moguće izvesti povratni skok i ispitivanje da li treba da se iskoči iz petlje.

Nakon formiranja novog rasporeda odgovoriti na sledeća pitanja:

1. Koliki je minimalan nivo softverske protočnosti?
2. Da li je petlja mogla imati drugačiju predpetlju i postpetlju od realizovanog rešenja i zašto?
3. Da li se mogla realizovati petlja bez pamćenja međurezultata u registrima ili FIFO redovima?

Napomene:

- Domaći zadatak se radi samostalno. Student treba da realizuje zadatak iz grupe koja se dobija na sledeći način: $gr = (brind \bmod 15) + 1$, gde je *brind* broj indeksa studenta (npr. student sa indeksom 2017/0895 realizuje zadatak iz grupe $11 = (895 \bmod 15) + 1$).
- Tačan datum i način predaje rešenja i odbrane će biti naknadno objavljeni.
- Za sva pitanja i nejasnoće u vezi sa postavkom domaćeg zadatka pisati na majav@etf.rs.

Grupa 1

Za zadatu petlju:

```
Do i = 3, 100
Op1: B(i) := B(i-1) * B(i-2)
Op2: E(i) := B(i-1) * A(i-1)
Op3: A(i) := B(i-1) + C(i-2)
Op4: C(i) := E(i) * D(i)
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovode do ulaza u drugu jedinicu za 1 ciklus. Množači/delitelji izvršavaju operacije za 3 ciklusa, a ALU jedinice za 1 ciklus. Pretpostaviti da su na raspolaganju FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasniti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 2

Za zadatu petlju:

```
Do i = 2, 100
Op1: A(i) := 17 * K(i)
Op2: B(i) := C(i) * D(i-1)
Op3: E(i) := A(i) + C(i)
Op4: D(i) := C(i) + K(i)
Op5: F(i) := E(i) + D(i)
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovode do ulaza u drugu jedinicu za 2 ciklusa. Množači/delitelji izvršavaju operacije za 2 ciklusa, a ALU jedinice za 1 ciklus. Pretpostaviti da su na raspolaganju FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasniti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 3

Za zadatu petlju:

```
Do i = 3, 100
Op1: A(i) := D(i) + B(i-2)
Op2: E(i) := A(i) * F(i-1)
Op3: C(i) := D(i) + A(i-2)
Op4: B(i) := D(i) + C(i)
Op5: F(i) := B(i) * E(i-1)
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovode do ulaza u drugu jedinicu za 2 ciklusa. Množači/delitelji izvršavaju operacije za 2 ciklusa, a ALU jedinice za 1 ciklus. Pretpostaviti da su na raspolaganju FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasniti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 4

Za zadatu petlju:

```
Do i = 3, 100
Op1: C(i) := A(i-1) + B(i-2)
Op2: A(i) := A(i-2) + D(i)
Op3: E(i) := F(i-1) * D(i)
Op4: B(i) := E(i) + A(i)
Op5: F(i) := C(i) * D(i)
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovode do ulaza u drugu jedinicu za 1 ciklus. Množači/delitelji izvršavaju operacije za 2 ciklusa, a ALU jedinice za 1 ciklus. Pretpostaviti da su na raspolaganju FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasniti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 5

Za zadatu petlju:

```
Do i = 3, 100
Op1: A(i) := B(i-2) + E(i)
Op2: B(i) := A(i) * D(i)
Op3: C(i) := F(i-1) * A(i)
Op4: F(i) := B(i) + C(i)
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovode do ulaza u drugu jedinicu za 2 ciklusa. Množači/delitelji izvršavaju operacije za 3 ciklusa, a ALU jedinice za 1 ciklus. Pretpostaviti da su na raspolaganju FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasnuti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 6

Za zadatu petlju:

```
Do i = 3, 100
Op1: A(i) := E(i) * K(i)
Op2: B(i) := A(i) + K(i)
Op3: C(i) := B(i-2) * E(i)
Op4: D(i) := C(i) + K(i)
Op5: F(i) := A(i) * D(i)
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovode do ulaza u drugu jedinicu za 2 ciklusa. Množači/delitelji izvršavaju operacije za 3 ciklusa, a ALU jedinice za 2 ciklusa. Pretpostaviti da su na raspolaganju i FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasnuti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 7

Za zadatu petlju:

```
Do i = 3, 100
Op1: A(i) := B(i) + K(i)
Op2: C(i) := B(i) * K(i)
Op3: D(i) := C(i) + G(i-1)
Op4: F(i) := A(i) * C(i)
Op5: G(i) := C(i) + D(i-2)
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovode do ulaza u drugu jedinicu za 1 ciklus. Množači/delitelji izvršavaju operacije za 3 ciklusa, a ALU jedinice za 2 ciklusa. Pretpostaviti da su na raspolaganju FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasnuti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 8

Za zadatu petlju:

```
Do i = 3, 100
Op1: A(i) := C(i-1) + B(i-2)
Op2: D(i) := B(i-1) * B(i-2)
Op3: B(i) := D(i) + E(i)
Op4: C(i) := D(i) * A(i)
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovode do ulaza u drugu jedinicu za 1 ciklus. Množači/delitelji izvršavaju operacije za 2 ciklusa, a ALU jedinice za 1 ciklus. Pretpostaviti da su na raspolaganju i FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasnuti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 9

Za zadatak petlju:

```
Do i = 1, 100
Op1: B(i) := A(i) + E(i)
Op2: D(i) := 11 * B(i)
Op3: C(i) := A(i) * E(i)
Op4: F(i) := A(i) + B(i)
Op5: G(i) := C(i) * D(i)
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovede do ulaza u drugu jedinicu za 2 ciklusa. Množači/delitelji izvršavaju operacije za 3 ciklusa, a ALU jedinice za 2 ciklusa. Pretpostaviti da su na raspolaganju FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasnuti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 10

Za zadatak petlju:

```
Do i = 4, 100
Op1: A(i) := E(i) * G(i)
Op2: B(i) := E(i) + G(i)
Op3: C(i) := A(i-3) + B(i-1)
Op4: D(i) := A(i) * B(i)
Op5: E(i) := A(i) * D(i)
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovede do ulaza u drugu jedinicu za 1 ciklus. Množači/delitelji izvršavaju operacije za 3 ciklusa, a ALU jedinice za 1 ciklus. Pretpostaviti da su na raspolaganju i FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasnuti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 11

Za zadatak petlju:

```
Do i = 3, 100  
Op1: B(i) := A(i) + D(i-2)  
Op2: C(i) := B(i) * E(i-1)  
Op3: E(i) := C(i) + D(i-1)  
Op4: D(i) := A(i) + B(i)  
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovode do ulaza u drugu jedinicu za 2 ciklusa. Množači/delitelji izvršavaju operacije za 2 ciklusa, a ALU jedinice za 1 ciklus. Pretpostaviti da su na raspolaganju FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasniti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 12

Za zadatak petlju:

```
Do i = 4, 100  
Op1: B(i) := A(i) * C(i-1)  
Op2: C(i) := A(i) * B(i)  
Op3: D(i) := F(i-3) + B(i)  
Op4: E(i) := A(i) + D(i)  
Op5: F(i) := C(i) + E(i)  
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovode do ulaza u drugu jedinicu za 1 ciklus. Množači/delitelji izvršavaju operacije za 3 ciklusa, a ALU jedinice za 1 ciklus. Pretpostaviti da su na raspolaganju i FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasniti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 13

Za zadatak petlju:

```
Do i = 1, 100
Op1: B(i) := A(i) * J(i)
Op2: C(i) := A(i) + J(i)
Op3: D(i) := B(i) * B(i)
Op4: E(i) := B(i) * C(i)
Op5: G(i) := B(i) + E(i)
Op6: F(i) := D(i) + G(i)
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovode do ulaza u drugu jedinicu za 1 ciklus. Množači/delitelji izvršavaju operacije za 3 ciklusa, a ALU jedinice za 2 ciklusa. Pretpostaviti da su na raspolaganju FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasniti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 14

Za zadatak petlju:

```
Do i = 3, 100
Op1: A(i) := B(i-1) + G(i)
Op2: C(i) := A(i) * B(i-2)
Op3: B(i) := E(i) + G(i)
Op4: D(i) := A(i) + C(i)
Op5: F(i) := A(i-1) * D(i)
End
```

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovode do ulaza u drugu jedinicu za 2 ciklusa. Množači/delitelji izvršavaju operacije za 3 ciklusa, a ALU jedinice za 1 ciklus. Pretpostaviti da su na raspolaganju i FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasniti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 15

Za zadatu petlju:

Do $i = 4, 100$

Op1: $B(i) := A(i-1) + F(i)$

Op2: $E(i) := B(i-3) * F(i)$

Op3: $C(i) := D(i-1) + E(i)$

Op4: $D(i) := D(i-1) + F(i)$

Op5: $A(i) := C(i) * D(i)$

End

formirati petlju softverske protočnosti koja se izvršava za minimalni broj ciklusa prema tipu petlje. Pretpostaviti neograničene hardverske resurse, uključujući registre. Takođe pretpostaviti da su sve funkcionalne jedinice međusobno povezane i da se rezultati jedne jedinice sprovode do ulaza u drugu jedinicu za 1 ciklus. Množači/delitelji izvršavaju operacije za 2 ciklusa, a ALU jedinice za 1 ciklus. Pretpostaviti da su na raspolaganju i FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasniti za jedan ciklus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.