DOMAĆI ZADATAK 2016/2017.

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. 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 mod 15) + 1, gde je brind broj indeksa studenta (npr. student sa indeksom 2017/0895 realizuje zadatak iz grupe 11 = (895 mod 15) + 1).
- Odbrana domaćih zadataka će biti organizovana početkom juna. 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 DoAll petlju:

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

napraviti petlju softverske protočnosti koja se izvršava za jedan ciklus. 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. Na raspolaganju su i FIFO memorije sa ulaznim mizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasniti za jedan cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 2

Za zadatu DoAccross petlju:

```
Do i = 4, 100
Op1:    C(i) := A(i-3) + B(i)
Op2:    X(i) := F(i-2) + C(i)
Op3:    A(i) := C(i) * D(i)
Op4:    F(i) := A(i) * X(i)
End
```

napraviti petlju softverske protočnosti koja se izvršava za najmanji broj ciklusa. Pretpostaviti neograničene hardverske resurse, uključujući registre. Na raspolaganju je i FIFO memorija sa ulaznim nizovima. 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. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasniti za jedan cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 3

Za zadatu DoAll petlju:

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

napraviti petlju softverske protočnosti koja se izvršava za jedan ciklus. 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 cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 4

Za zadatu DoAccross petlju:

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

napraviti petlju softverske protočnosti koja se izvršava za najmanji broj ciklusa. 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 i FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasniti za jedan cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 5

Za zadatu DoAll petlju:

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

napraviti petlju softverske protočnosti koja se izvršava za jedan ciklus. 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 cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 6

Za zadatu DoAcross petlju:

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

napraviti petlju softverske protočnosti koja se izvršava za najmanji broj ciklusa. 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 i FIFO memorije sa ulaznim nizovima. Pamćenje rezultata se može raditi u pomoćnim registrima kada rezultat treba zakasniti za jedan cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 7

Za zadatu DoAll petlju:

```
Do i = 1, 100
Op1:    B(i) := A(i) * X(i)
Op2:    C(i) := A(i) * M(i)
Op3:    D(i) := C(i) + C(i)
Op4:    F(i) := B(i) + C(i)
End
```

napraviti petlju softverske protočnosti koja se izvršava za jedan ciklus. 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 zakasniti za jedan cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 8

Za zadatu DoAccross petlju:

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

napraviti petlju softverske protočnosti koja se izvršava za najmanji broj ciklusa. 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 cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 9

Za zadatu DoAll petlju:

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

napraviti petlju softverske protočnosti koja se izvršava za jedan ciklus. 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 cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 10

Za zadatu DoAccross petlju:

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

napraviti petlju softverske protočnosti koja se izvršava za najmanji broj ciklusa. 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 cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 11

Za zadatu DoAll petlju:

```
Do i = 1, 100
Op1:    A(i) := X(i) * X(i)
Op2:    C(i) := X(i) + X(i)
Op3:    D(i) := A(i) + C(i)
Op4:    F(i) := A(i) * D(i)
End
```

napraviti petlju softverske protočnosti koja se izvršava za jedan ciklus. 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 cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 12

Za zadatu DoAccross petlju:

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

napraviti petlju softverske protočnosti koja se izvršava za najmanji broj ciklusa. 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 cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 13

Za zadatu DoAll petlju:

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

napraviti petlju softverske protočnosti koja se izvršava za jedan ciklus. 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 cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 14

Za zadatu DoAccross petlju:

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

napraviti petlju softverske protočnosti koja se izvršava za najmanji broj ciklusa. 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 cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.

Grupa 15

Za zadatu DoAll petlju:

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

napraviti petlju softverske protočnosti koja se izvršava za jedan ciklus. 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 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 cilkus. Ako je potrebno kašnjenje veće od jednog ciklusa, koristi se FIFO sa protočnim registrima na krajevima.