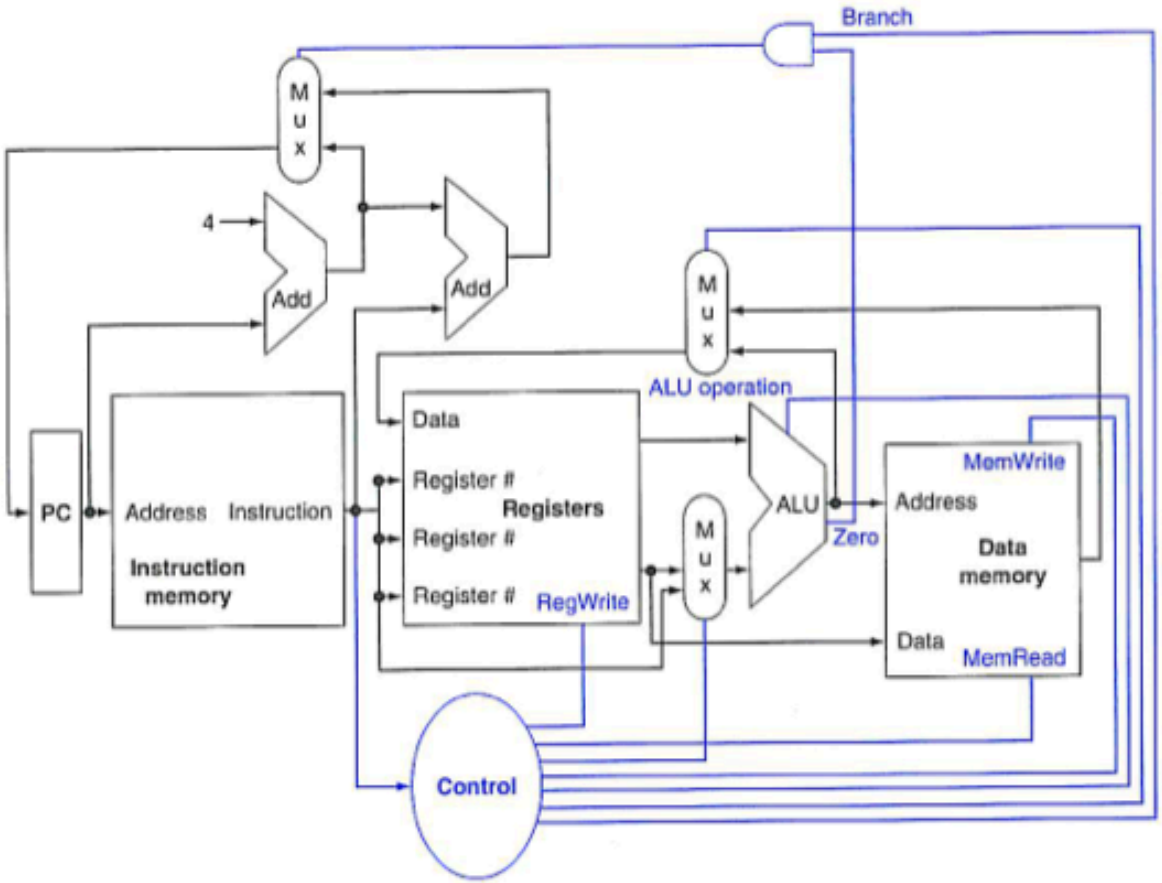
PROBLEME

1. Diferite instrucțiuni utilizează diferite blocuri hardware în cazul implementării bazate pe un singur ciclu de ceas. Se consideră următoarele instrucțiuni:

add Rd, Rs, Rt ⇔ Reg[Rd] = Reg[Rs] + Reg[Rt]
lw Rt, offs(Rs) ⇔ Reg[Rt] = Mem[Reg[Rs] + Offs]

a) Care sunt valorile semnalelor de control generate confor figurii de mai jos



b) Ce resurse sunt folosite pentru realizarea instrucțiunilor ?

c) Ce resurse produc o ieșire care nu este utilizată de aceste instrucțiuni ? Ce resurse nu produc nici o ieșire ?

2. În tabelul de mai jos sunt prezentați timpii de execuție necesari pentru anumite blocuri funcționale. Care este calea critică pentru realizarea instrucțiunii AND ?

	I-Mem	Add	Mux	ALU	Regs	D-Mem	Control
a.	400ps	100ps	30ps	120ps	200ps	350ps	100ps
b.	500ps	150ps	100ps	180ps	220ps	1000ps	65ps

```
i1:  add  r1, r2, r3
i2:  lw   r4, 0(r1)
i3:  sw   12(r1), r4
```

RAW hazard – read after write; r1 este citit de o instrucțiune care vine după o instrucțiune care scrie r1

```
i1:  add   r3, 4(r4)  - un operand in registru și cel de al doilea în memorie
i2:  mov  r3, r5
```

WAW hazard – write after write; i2 se termină înaintea lui i1 => de fapt execuție inversă a operațiilor.

```
i1:  mov   20(r3), 30(r4)  //  este o mutare în memorie, necesită calcularea a 2
adrese efective.
i2:  add    r3, 1              //  incrementare r3
```

WAR hazard – write after read; adresa efectivă a primului operand nu este determinată până când valoarea lui r3 a fost incrementată de i2 => valoare incorectă pentru adresa efectivă

```
i1:  lw   r1, 0(r2)
i2:  sub  r4, r1, r5
i3:  and  r6, r1, r7
i4:  or   r8, r1, r9
```

Această tehnică se numește *pipeline interlock*.

3. Specificați dacă existe hazarde în următoarele secvențe de cod și locul apariției lor:

```
P11: lw $1, 40($6)          P21: lw $5, -16($5)
P12: add $6, $2, $2         P22: sw $5, -16($5)
P13: sw $6, 50($t1)         P23: add $5, $5, $5
```

4. Eliminați hazardele din secvențele următoare de cod dacă exista:

```
P11: lw $1, 40($6)          P21: lw $5, -16($5)
P12: add $6, $2, $2         P22: sw $5, -16($5)
P13: sw $6, 50($t1)         P23: add $5, $5, $5
```

5. Se consideră următoarele secvențe de cod:

```
P11: lw $1, 40($6)
P12: beq $2,$0, Label:   pp $2=$0
P13: sw $6, 50($2)
Label: add $2, $3,$4
       sw $3, 50($4)
```

```
P21: lw $5, -16($5)
P22: sw $4, -16($4)
P23: lw $3, -20($4)
P24: beq $2, $0 Label: pp $2 != $0
P25: add $5, $1, $4
```

Se presupune că toate salturile sunt perfect predictibile (nu avem hazarde structurale). Dacă avem o singură memorie (instrucțiuni și date) există un hazard structural ori de câte ori citim o instrucțiune în același ciclu de ceas în care o instrucțiune accesează datele. Acest tip de hazard va fi rezolvat întotdeauna în favoarea instrucțiunii care accesează date. Care este timpul total de execuție a instrucțiunilor prin banda de asamblare presupunând că avem o singură memorie și că banda de asamblare are 5 stagii

Se pot adăuga nop-uri pentru soluționarea hazardelor în acest caz ?

6. Se consideră următoarea secvență de cod:

```
lw      $1, 40($6)
add     $5, $5, $5
```

a) În cadrul execuției acestor instrucțiuni ce se menține în fiecare registru de pipe?

b) Care registre sunt necesare la citire și care sunt citite ?

c) Ce se realizează în cadrul stagiilor EX și MEM ?

3. Specificați dacă existe hazarde în următoarele secvențe de cod și locul apariției lor:

```
P11: lw $1, 40($2)          P21: add $1, $2, $3
P12: add $2, $3, $3         P22: sw $2, 0($1)
P13: add $1, $1, $2         P23: lw $1, 4($2)
P14: sw $1, 20($2)          P24: add $2, $2, $1
```

b) Să se găsească toate hazardele într-un pipe cu 5 stagii daca există/nu există forwarding

c) Presupunem că înintea execuției oricărei instrucțiuni, toate valorile din memorie sunt 0 iar valorile registrelor $0, $1, $2 și $3 sunt următoarele:

```
P1  0   1   31  1000
P2  0   -2  63  2500
```

Care valoare va fi forward-ată prima și care este valoarea ei ?

d) Prezentați secvența de program ce va fi executată de către procesor