# PROBLEMES 1: Optimització Processadors (RISC-V)

## 1.1 Penseu en la següent instrucció:

Instrucció: and rd, rs1, rs2
Interpretació: Reg [rd] = Reg [rs1] AND Reg [rs2]

1. Quins són els valors dels senyals de control generats pel bloc "control" de la figura 1 per a aquesta instrucció? Determineu el guany en velocitat d'un cas comparada amb l'altre. Doneu el resultat en %.
2. Quins recursos (blocs) fan una funció útil per a aquesta instrucció?
3. Quins recursos (blocs) no produeixen cap sortida per a aquesta instrucció? Quins recursos produeixen una producció que no s'utilitza?

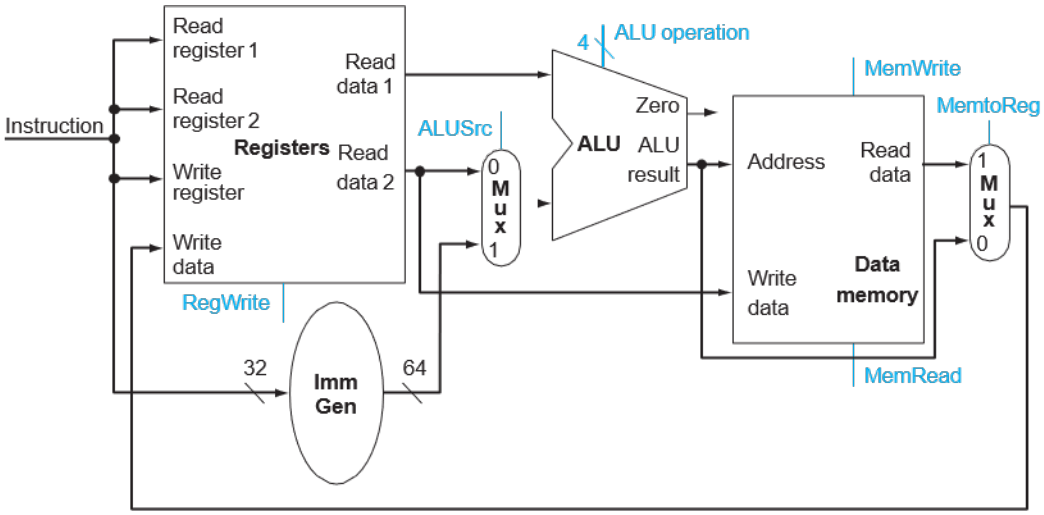
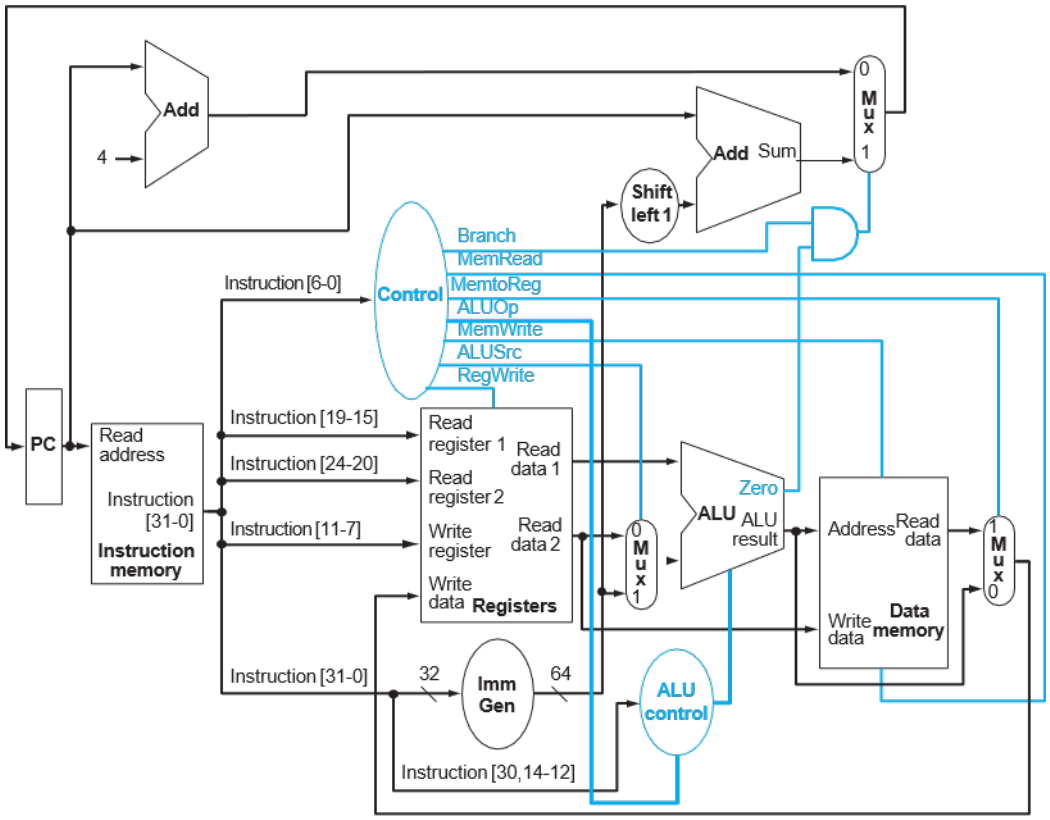



Figura 1: "Datapath"

1.2 Expliqueu cadascun dels “no m’importa” (“X”) de la figura 2b.




(a)

| Instruction | ALUSrc | Memto-Reg | Reg-Write | Mem-Read | Mem-Write | Branch | ALUOp1 | ALUOp0 |
|---|---|---|---|---|---|---|---|---|
| R-format | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| ld | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| sd | 1 | X | 0 | 0 | 1 | 0 | 0 | 0 |
| beq | 0 | X | 0 | 0 | 0 | 1 | 0 | 1 |

(b)

Figura 2: (a) “Datapath” simplificat amb la unitat de control del RISC-V single cycle.
(b) Taula amb els valors de les línies de control segons tipus d’instrucció.

1.3 Penseu en la combinació d'instruccions següent:

| R-type | I-type (non-ld) | Load | Store | Branch | Jump |
|---|---|---|---|---|---|
| 24% | 28% | 25% | 10% | 11% | 2% |

1. Quina fracció de totes les instruccions utilitza la memòria de dades?
2. Quina fracció de totes les instruccions utilitza la memòria d'instruccions?
3. Quina fracció de totes les instruccions utilitzen l'extensió de signe?
4. Què fa l'extensió de signe durant cicles en què no és necessària la seva sortida?

1.4 Quan es fabriquen xips de silici, els defectes de materials (per exemple, el silici) i els errors de fabricació poden provocar circuits defectuosos. Un defecte molt comú és que la línia d'una senyal es "trenqui" i registri sempre un 0 lògic, sovint s'anomena un error "stuck at 0". Ajudeu-vos de la figura 1.

1. Quines instruccions no funcionen correctament si la senyal MemToReg està "stuck at 0"?
2. Quines instruccions no funcionen correctament si la línia ALUSrc està "stuck at 0"?

1.5 En aquest exercici, examinarem detalladament com s'executa una instrucció en un "datapath" d'un cicle únic (single cycle). Els problemes d'aquest exercici fan referència al cicle de rellotge en què el processador obté la següent instrucció: 0x00c6ba23 (ISA RV64I).

1. Quins són els valors de les entrades de la unitat de control ALU per a aquesta instrucció?
2. Quina és la nova adreça del PC després d'executar aquesta instrucció? Ressalteu el camí a través del qual es determina aquest valor (figura 2).
3. Per a cada mux, mostreu els valors de les seves entrades i sortides durant l'execució d'aquesta instrucció. Enumereu els valors que són sortides de registre com Reg [xn] (on n és el nombre de reg).
4. Quins són els valors d'entrada per a l'ALU i les dues unitats addicionals?

1.6 Afegiu tantes instruccions “NOOP” com creieu al codi següent per tal que pugui funcionar en un pipeline de 5 etapes del processador RISC-V sense “Forward Unit”. Recordeu que tenim 32 registres que van des de l’x0 fins a l’x31.

| Programa | Execució amb NOOPs |
|---|---|
| Addi x11, x12, 5 | |
| Add x13, x11, x12 | |
| Addi x14, x11, 15 | |
| Add x15, x11, x11 | |
| | |
| | |
| | |
| | |
| | |
| | |

1.7 En un processador RISC-V de 5 etapes examinem com afecta el pipeline al temps de cicle del rellotge del processador. Les qüestions d’aquest exercici suposen que cada etapa triga un temps diferent en executar-se, en particular els temps d’execució de cada etapa són els següents:

| IF | ID | EX | MEM | WB |
|---|---|---|---|---|
| 250 ps | 350 ps | 150 ps | 300 ps | 200 ps |

Suposeu també que les instruccions executades pel processador es desglossen de la manera següent:

| ALU/Logic | Jump/Branch | Load | Store |
|---|---|---|---|
| 45% | 20% | 20% | 15% |

1. Quin és el temps de cicle de rellotge mínim (en ps) en un processador amb pipeline i en un sense pipeline ?
2. Quant temps triga en executar-se una instrucció de tipus Load en un processador amb pipeline i en un sense pipeline ?
3. Si podem dividir una etapa del pipeline en dues noves etapes, cadascuna trigarà la meitat de temps en executar-se que l’etapa

original, quina etapa dividiríeu i quin és el nou temps de cicle del rellotge del processador amb pipeline?

4. Suposant que no hi ha “stalls” ni “hazards”, quina és percentatge d’utilització de la memòria de dades per a les instruccions executades pel processador amb pipeline?
5. Suposant que no hi ha “stalls” ni “hazards”, quina és percentatge d’utilització del port d’escriptura dels Registres per a les instruccions executades pel processador amb pipeline?

## 1.8 Assumiu un processador RISC-V amb 5 etapes de pipeline.

1. Calculeu el temps de computació si en el processador **NO** tenim Unitat de Forwarding Paths. Ompliu la taula emprant “stalls” per solucionar les possibles dependències de dades. (aquest codi pot requerir més o menys cicles dels tabulats)

| | **Temps →** | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Instrucció** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** |
| sd x29, 12(x16) | | | | | | | | | | | | | | | | | | | | | | | |
| ld x29, 8(x16) | | | | | | | | | | | | | | | | | | | | | | | |
| sub x17, x9, x14 | | | | | | | | | | | | | | | | | | | | | | | |
| beqz x17, label | | | | | | | | | | | | | | | | | | | | | | | |
| add x15, x17, x14 | | | | | | | | | | | | | | | | | | | | | | | |
| add x16, x15, x14 | | | | | | | | | | | | | | | | | | | | | | | |

2. Calculeu el temps de computació si en el processador **SÍ** tenim Unitat de Forwarding Paths. Ompliu la taula emprant "stalls" per solucionar les possibles dependències de dades. (aquest codi pot requerir més o menys cicles dels tabulats)

| | **Temps →** | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Instrucció** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** |
| sd x29, 12(x16) | | | | | | | | | | | | | | | | | | | | | | | |
| ld x29, 8(x16) | | | | | | | | | | | | | | | | | | | | | | | |
| sub x17, x9, x14 | | | | | | | | | | | | | | | | | | | | | | | |
| beqz x17, label | | | | | | | | | | | | | | | | | | | | | | | |
| add x15, x17, x14 | | | | | | | | | | | | | | | | | | | | | | | |
| add x16, x15, x14 | | | | | | | | | | | | | | | | | | | | | | | |

1.9 Assumim que el 20% de les instruccions executades per un programa són de tipus "branch". No hi ha "stalls" deguts a dependència de dades. Tenim predicció estàtica de salt del tipus "no es compleix". Els temps "perdut" (penalty) és d'1 cicle de rellotge:

1. Determineu el temps d'execució dels dos casos següents: en el 30% dels "branch" la condició es compleix. En el 70 % dels "branch" la condició es compleix.
2. Determineu el guany en velocitat d'un cas comparada amb l'altre. Doneu el resultat en %.

1.10 La importància de tenir un bon predictor de “braches” depèn de la freqüència amb què s’executen els “branches” condicionals. Si també considerem la precisió del pronòstic dels “branches”, tots dos conceptes determinaran la quantitat de temps que el processador es passa aturat a causa de “branches” imprevistos. En aquest exercici, suposem que el desglossament d’instruccions en diverses categories d’instruccions és el següent:

| ALU/Logic | Branch | Load | Store |
|---|---|---|---|
| 40% | 25% | 25% | 10% |

Assumeix també les precisions dels predictors de següents:

| Always-Taken | Always-Not-Taken |
|---|---|
| 45% | 55% |

Suposeu que fem servir un processador RISC-V de 5 etapes, on els “branches” es resolen al final de l’etapa d’execució. Suposeu, també, que el programa és infinit i partim d’un CPI ideal (no tenim hazards deguts a dades).

1. Quin és l’increment en el CPI degut a la mala predicció del predictor “Always-Taken”?
2. Quin és l’increment en el CPI degut a la mala predicció del predictor “Always-Not-Taken”?
3. L’ús d’una solució dinàmica, com una taula de d’història de salts, milloraria les prestacions ? Compara-ho amb els dos predictors estàtics emprats en aquesta qüestió.

1.11 Un processador superescalar com el de la Figura 3 ha d’executar les següents instruccions:

| | |
|---|---|
| addi | x2, x3, #100 |
| lw | x5,16(x6) |
| sub | x7, x8, x9 |
| sw | x10, 24(x11) |

Suposant que tots els registren ja contenen els valors, dibuixeu un diagrama mostrant el flux d’instruccions dins del pipeline.

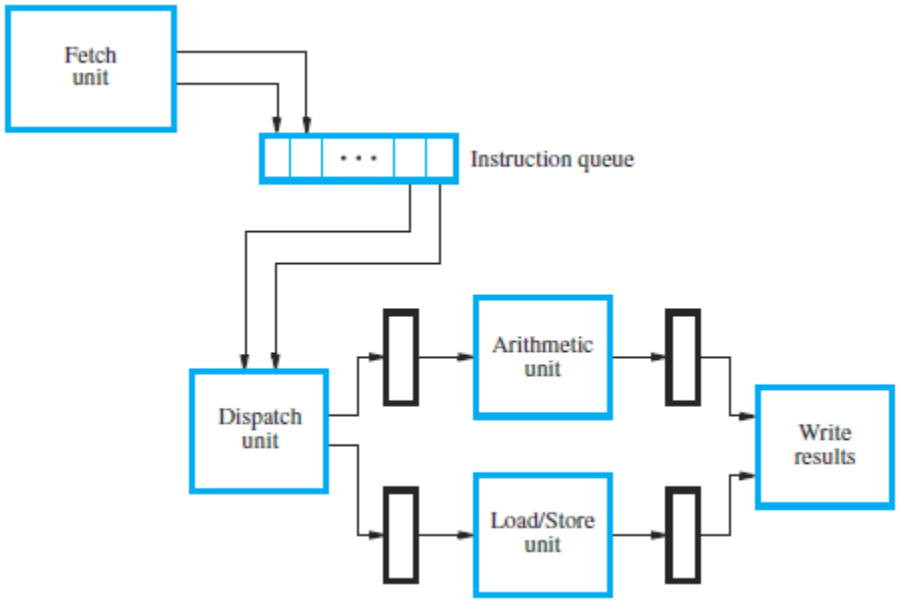



Figura 3: Esquemàtic del processador superescalar

1.12 Considereu un programa que consisteix en 4 accessos a memòria i 4 instruccions aritmètiques. Assumiu que no hi ha cap risc de dependència de dades entre les instruccions. Dues versions d'aquest programa s'executaran en un processador superescalar com el de la Figura 3. La primera versió del programa executa els 4 accessos a memòria consecutivament i després realitza les operacions aritmètiques. La segona versió del programa té les instruccions de memòria intercalades amb les aritmètiques. Realitza dos diagrames, multicicle per a comparar les dues versions del programa.

1.13 Assumiu que un programa no conté instruccions tipus "branch". El programa s'executa en un processador superescalar com el de la Figura 3. Calculeu quin és el temps d'execució si tenim una mescla d'instruccions que consisteix en un 75% instruccions aritmètiques i un 25% instruccions d'accés a memòria ? Quin és el guany en velocitat si comparem aquest processador amb un com el de la Figura 1 (ambdós utilitzen el mateix rellotge)?

1.14 Repetiu el problema 1.12 tenint en consideració que el nou programa conté un 15% d'instruccions tipus "branch" on la condició mai es compleix, 65% d'instruccions aritmètiques i 20% d'instruccions d'accés a memòria. Suposeu una precisió de predicció del 100% en totes les instruccions tipus "branch"