Teoricopràctic 5

Introducció als ordinadors

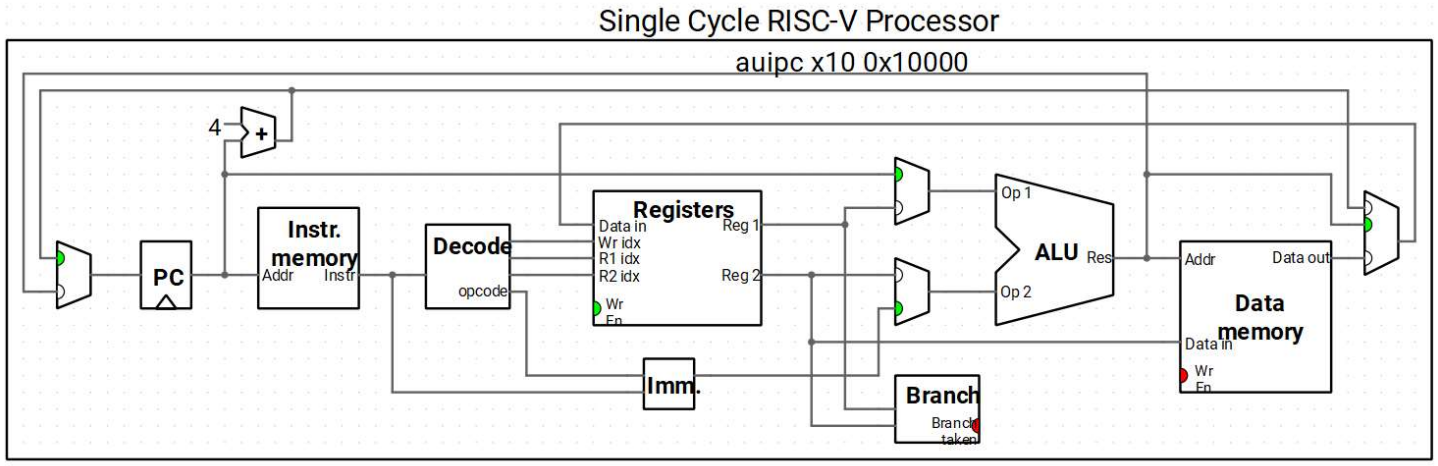
Funcionament de la CPU. Relacionant el camí de dades amb la unitat de control

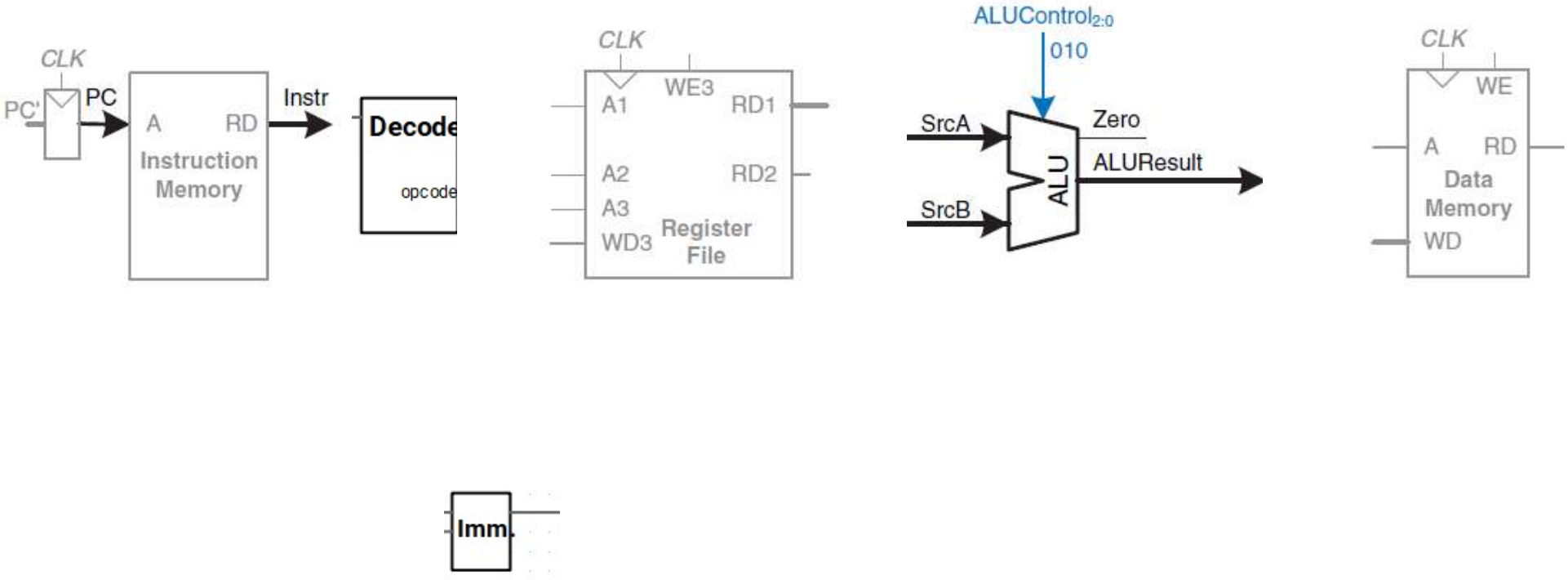
- Programa de test

```
1 .data
2 xx: .word 4,5
3 .text
4 la a0, xx
5 lw a1, 0(a0)
6 lw a2, 4(a0)
7 add a3, a2, a1
8 la a4, xx
9 sw a3, 8(a0)
```

```
0:      10000517        auipc x10 0x10000
4:      00050513        addi x10 x10 0
8:      00052583        lw x11 0(x10)
c:      00452603        lw x12 4(x10)
10:     00b606b3        add x13 x12 x11
14:     10000717        auipc x14 0x10000
18:     fec70713        addi x14 x14 -20
1c:     00d52423        sw x13 8(x10)
```

- Micro escollit




CLK
PC'
PC
A
RD
Instruction Memory
Instr
Decode
opcode
CLK
WE3
A1
RD1
A2
RD2
A3
WD3
Register File
ALUControl2:0
010
SrcA
SrcB
ALU
Zero
ALUResult
CLK
WE
A
RD
Data Memory
WD
Imm

Primera instrucció

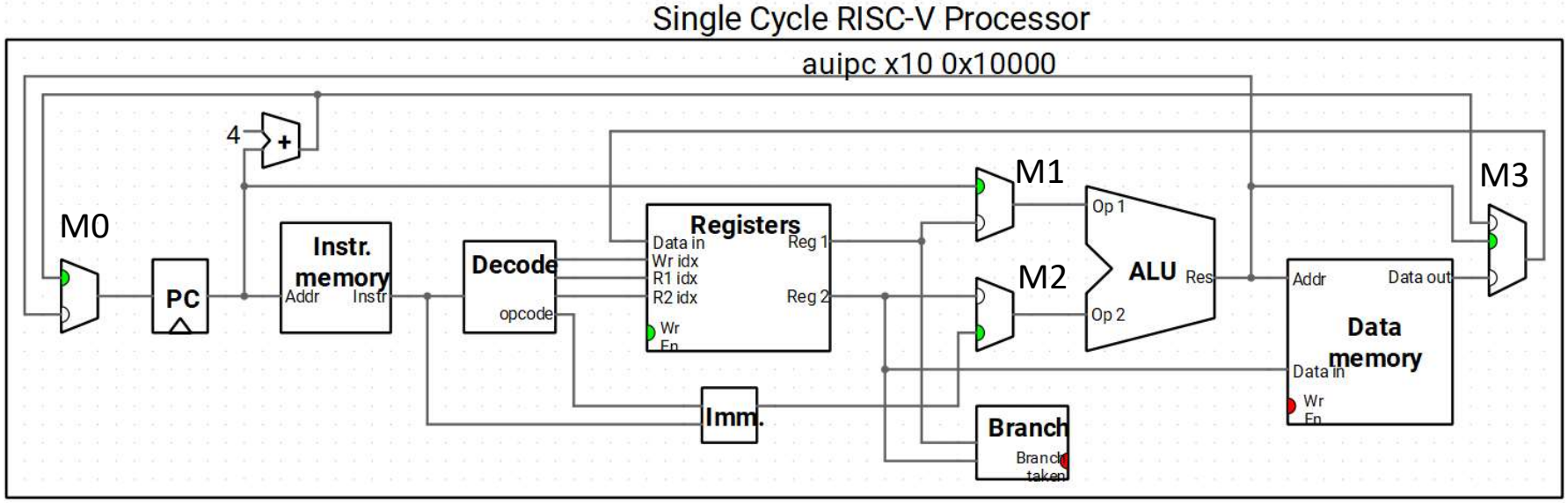
- PC<= PC+4 Enable Write
- X10 <= PC+(Imm <<12 bits)
- ALU <= ADD
- M0 <=0
- M1 <=1
- M2 <=0
- M3<=01
- DWR<=A
- DLR1 <=x
- DLR2 <=x

auipc rd, immediate `x[rd] = pc + sext(immediate[31:12] << 12)`

Add Upper Immediate to PC. Tipo U, RV32I y RV64I.

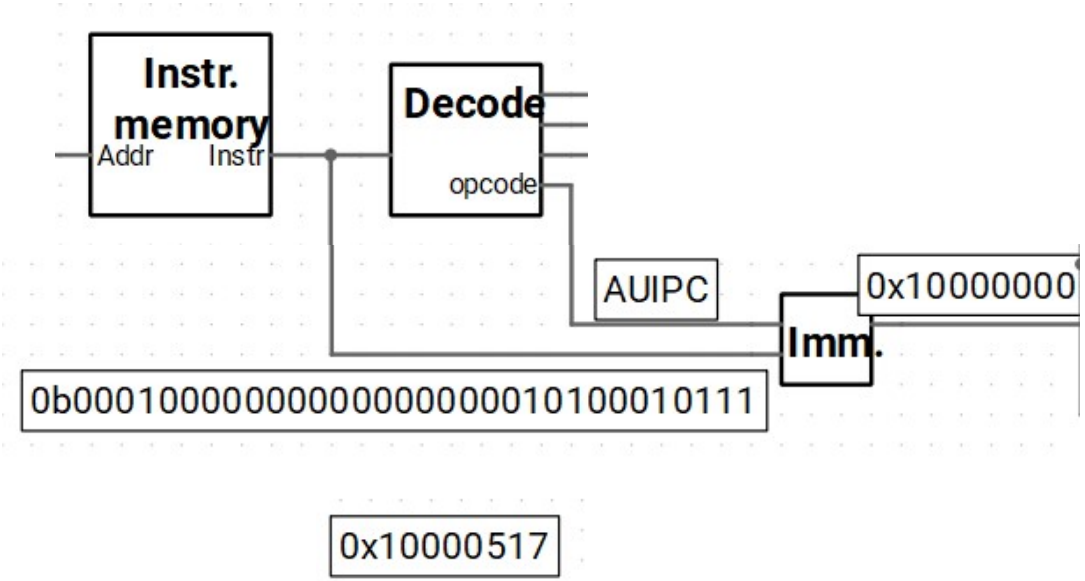
Suma el *inmediato* sign-extended de 20 bits, corrido a la izquierda por 12 bits, al *pc*, y escribe el resultado en x[*rd*].

31 — 12	11 — 7	6 — 0
immediate[31:12]	rd	0010111



Primera instrucció

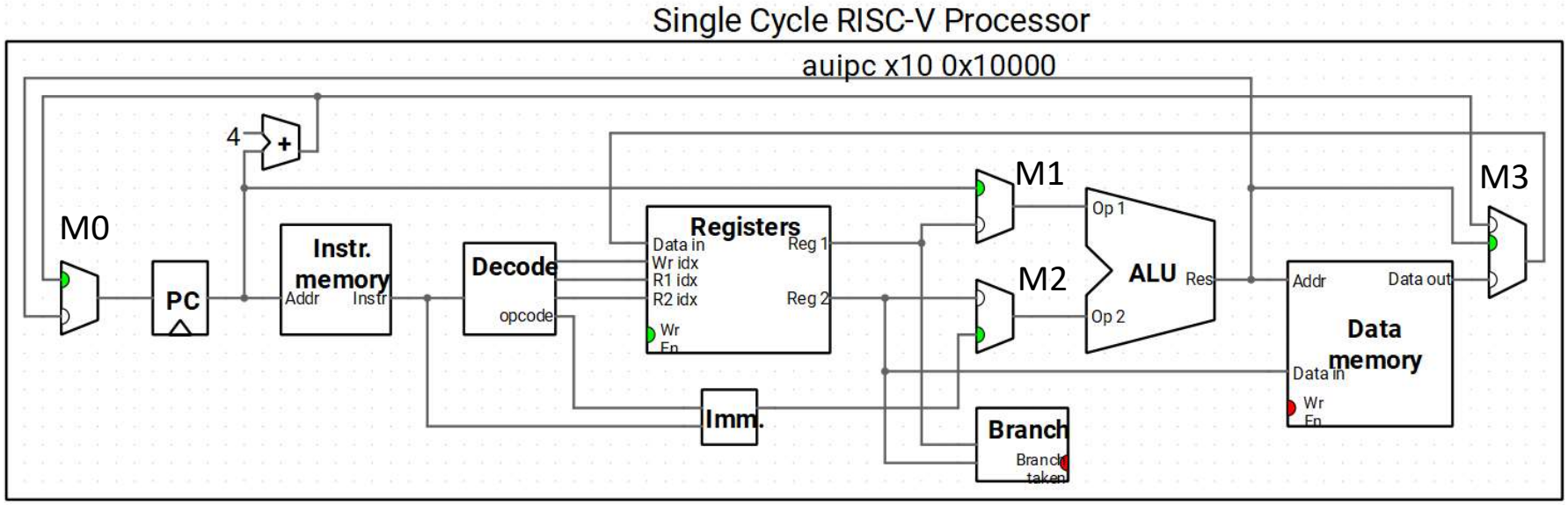
- És interessant observar com es genera el valor immediat



Primera instrucció

- Per M1 tenim el [PC]
- Per M2 tenim el valor de l'immediat <<12 bits

x10	a0	0x10000000



Primera instrucció

- Exercici:

Indiqueu quines són les entrades que ha de tenir la U.C. i quines són les sortides per executar aquesta instrucció

Segona instrucció

- PC<= PC+4 Enable Write
- X10 <= [X10] + Imm
- ALU <= ADD
- M0 <=0
- M1 <=0
- M2 <=0
- M3<=01
- DWR<=A
- DLR1 <=A
- DLR2 <=x

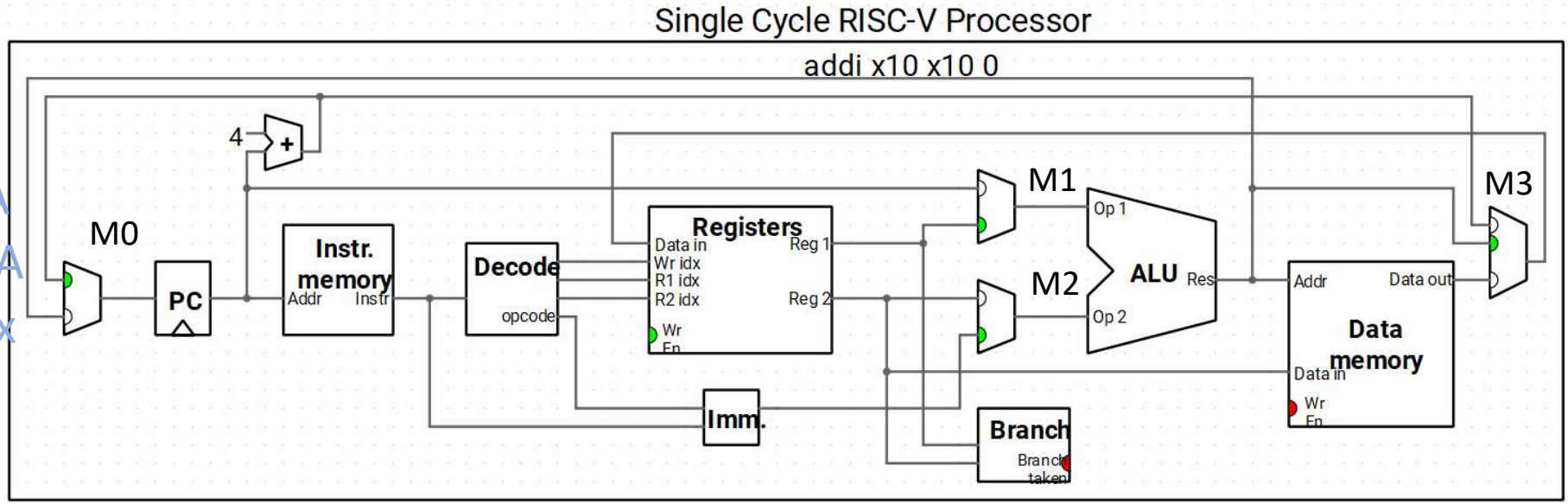
addi rd, rs1, immediate `x[rd] = x[rs1] + sext(immediate)`

Add Immediate. Tipo I, RV32I y RV64I.

Suma el *inmediato* sign-extended al registro x[*rs1*] y escribe el resultado en x[*rd*]. Overflow aritmético ignorado.

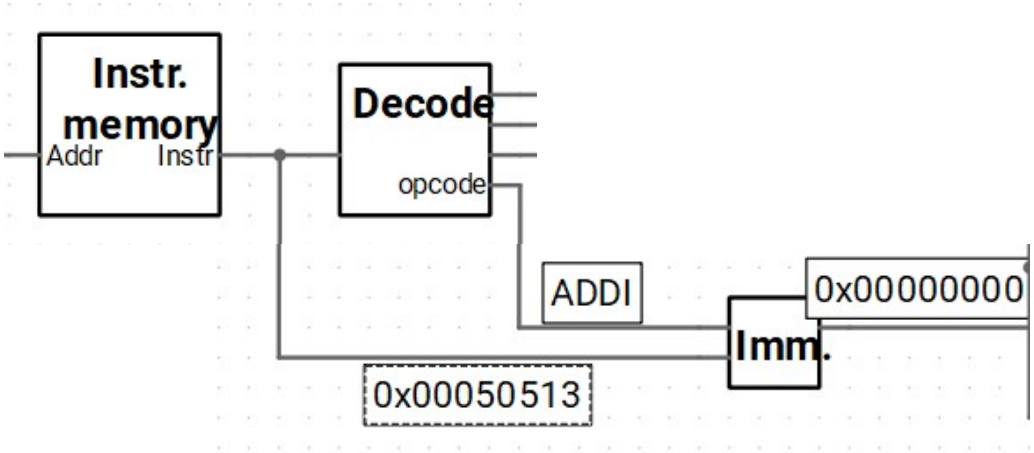
Formas comprimidas: **c.li** rd, imm; **c.addi** rd, imm; **c.addi16sp** imm; **c.addi4spn** rd, imm

31 — 20	19 — 15	14 — 12	11 — 7	6 — 0
immediate[11:0]	rs1	000	rd	0010011



Segona instrucció

- Observeu ara com varia la sortida de l'immediat



- En aquest cas, no hi ha desplaçament del valor, però s'esten al valor del signe => 0x000 (12 bits) passa a 0x00000000

Segona instrucció

- Exercici:

Indiqueu quines són les entrades que ha de tenir la U.C. i quines són les sortides per executar aquesta instrucció

Tercera instrucció

- PC<= PC+4 Enable Write
- X11 <= [@([X10] + Imm)]
- ALU <= ADD
- M0 <=0
- M1 <=0
- M2 <=0
- M3<=01
- DWR<=B
- DLR1 <=A
- DLR2 <=x

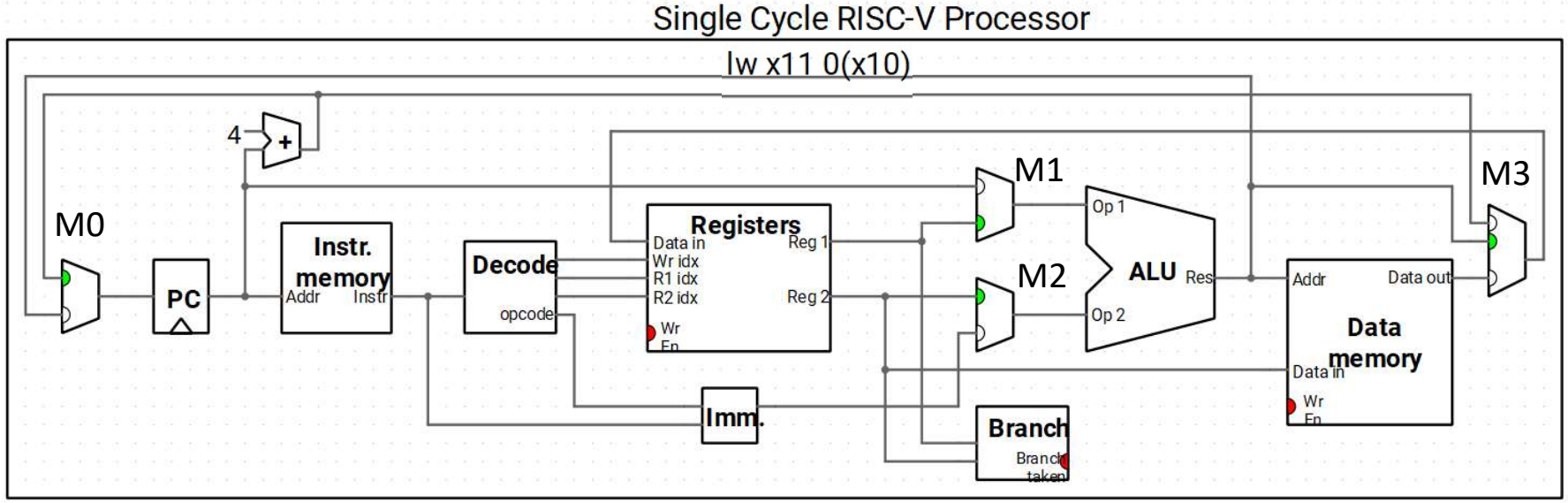
lw rd, offset(rs1) `x[rd] = sext(M[x[rs1] + sext(offset)][31:0])`

Load Word. Tipo I, RV32I y RV64I.

Carga cuatro bytes de memoria en la dirección x[*rs1*] + *sign-extend*(*offset*) y los escribe en x[*rd*]. Para RV64I, el resultado es extendido en signo.

Formas comprimidas: **c.lwsp** rd, offset; **c.lw** rd, offset(rs1)

31 — 20	19 — 15	14 — 12	11 — 7	6 — 0
offset[11:0]	rs1	010	rd	0000011



Tercera instrucció

lw rd, offset(rs1) `x[rd] = sext(M[x[rs1] + sext(offset)][31:0])`

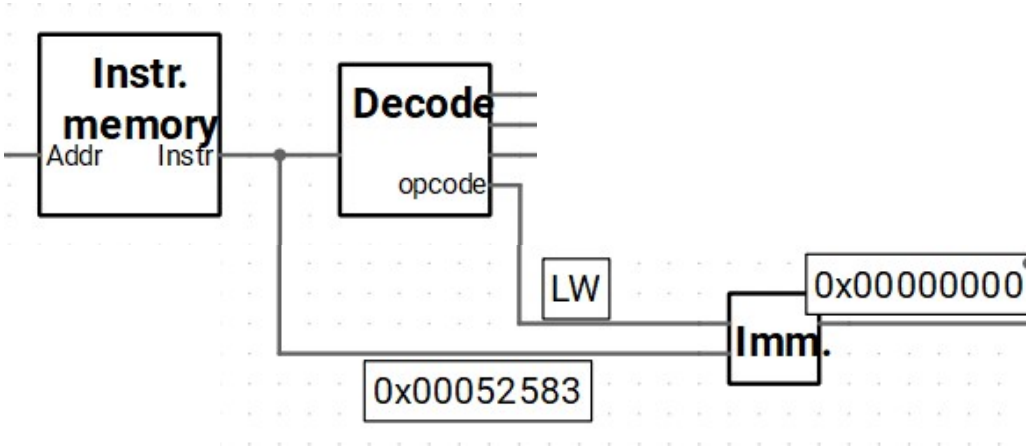
Load Word. Tipo I, RV32I y RV64I.

Carga cuatro bytes de memoria en la dirección x[*rs1*] + *sign-extend*(*offset*) y los escribe en x[*rd*]. Para RV64I, el resultado es extendido en signo.

Formas comprimidas: **c.lwsp** rd, offset; **c.lw** rd, offset(rs1)

31 — 20	19 — 15	14 — 12	11 — 7	6 — 0
offset[11:0]	rs1	010	rd	0000011

- Observeu ara com varia la sortida de l'immediat



- En aquest cas, no hi ha desplaçament del valor, però s'esten al valor del signe => 0x000 (12 bits) passa a 0x00000000

Tercera instrucció

- Exercici:

Indiqueu quines són les entrades que ha de tenir la U.C. i quines són les sortides per executar aquesta instrucció

Quarta instrucció

- PC<= PC+4 Enable Write
- X12 <= [@([X10] + Imm)]
- ALU <= ADD
- M0 <=0
- M1 <=0
- M2 <=0
- M3<=01
- DWR<=C
- DLR1 <=A
- DLR2 <=x

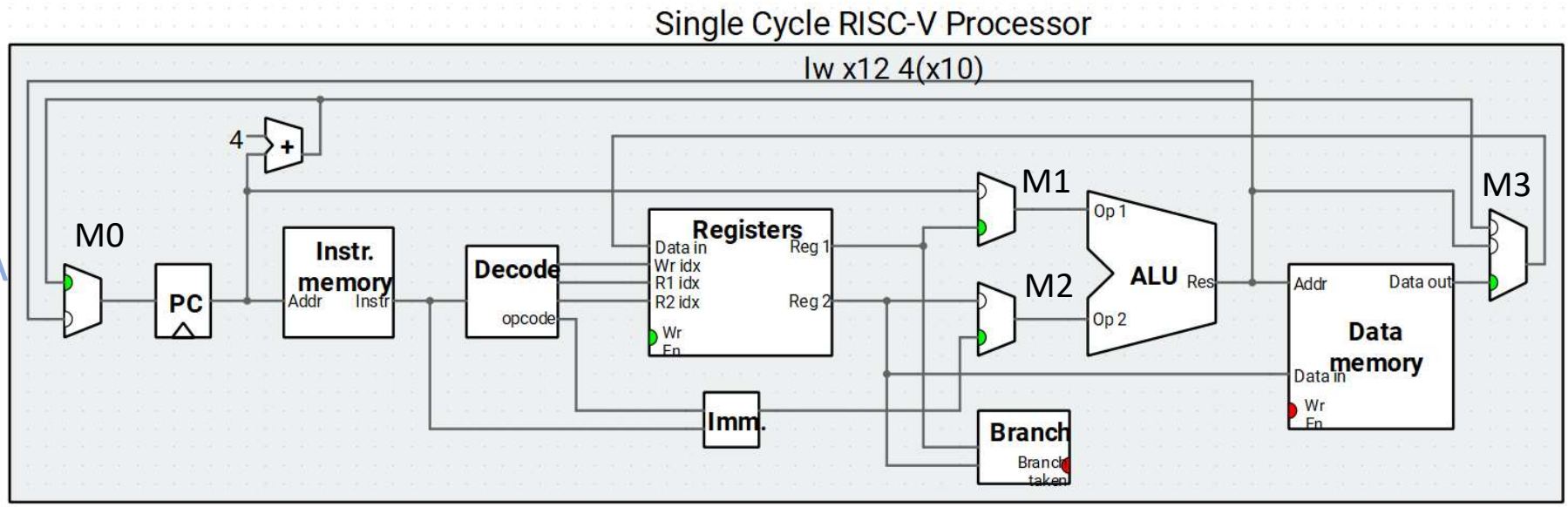
lw rd, offset(rs1) `x[rd] = sext(M[x[rs1] + sext(offset)][31:0])`

Load Word. Tipo I, RV32I y RV64I.

Carga cuatro bytes de memoria en la dirección x[*rs1*] + *sign-extend*(*offset*) y los escribe en x[*rd*]. Para RV64I, el resultado es extendido en signo.

Formas comprimidas: **c.lwsp** rd, offset; **c.lw** rd, offset(rs1)

31 — 20	19 — 15	14 — 12	11 — 7	6 — 0
offset[11:0]	rs1	010	rd	0000011



Quarta instrucció

lw rd, offset(rs1) `x[rd] = sext(M[x[rs1] + sext(offset)][31:0])`

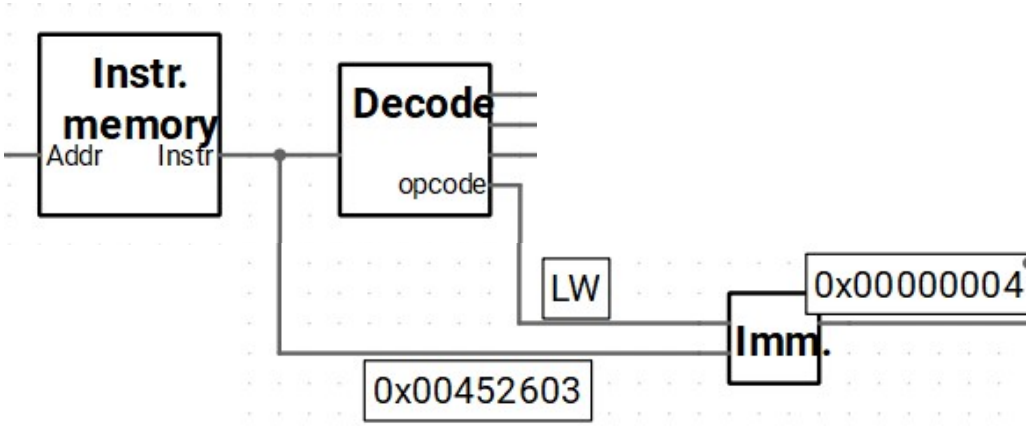
Load Word. Tipo I, RV32I y RV64I.

Carga cuatro bytes de memoria en la dirección x[*rs1*] + *sign-extend*(*offset*) y los escribe en x[*rd*]. Para RV64I, el resultado es extendido en signo.

Formas comprimidas: **c.lwsp** rd, offset; **c.lw** rd, offset(rs1)

31 — 20	19 — 15	14 — 12	11 — 7	6 — 0
offset[11:0]	rs1	010	rd	0000011

- Observeu ara com varia la sortida de l'immediat



- En aquest cas, no hi ha desplaçament del valor, però s'esten al valor del signe => 0x004 (12 bits) passa a 0x00000004

Cinquena instrucció

- PC<= PC+4 Enable Write
- X13 <= [X11] + [X12]
- ALU <= ADD
- M0 <=0
- M1 <=0
- M2 <=1
- M3<=01
- DWR<=D
- DLR1 <=B
- DLR2 <=C

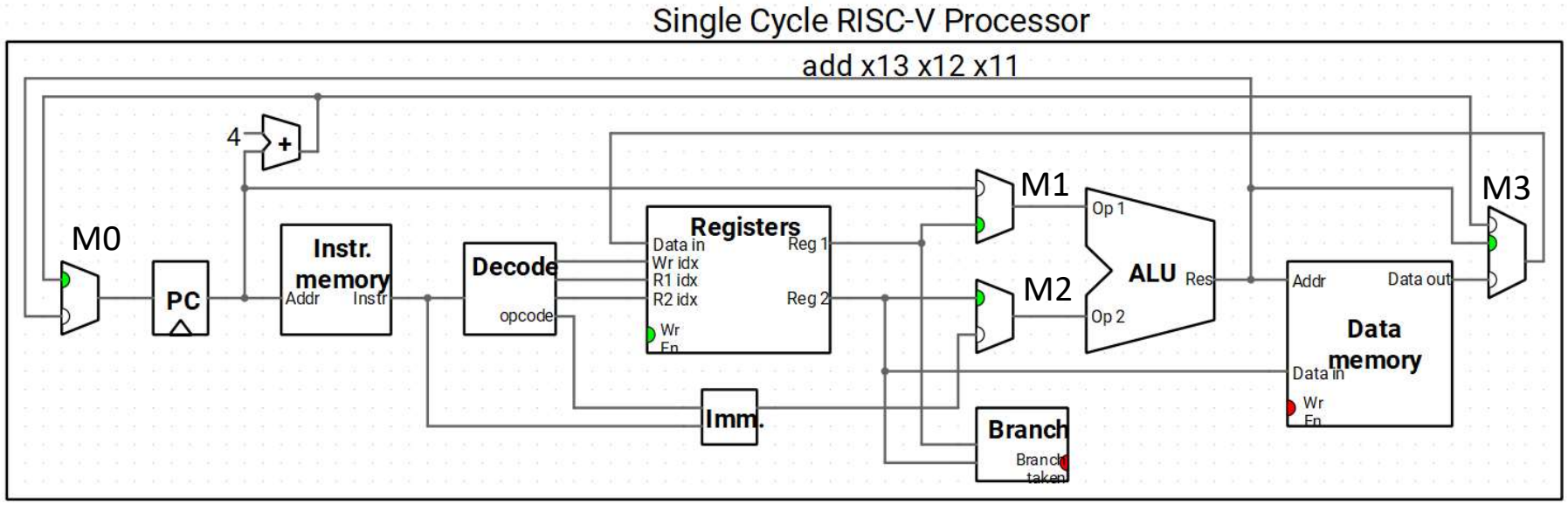
add rd, rs1, rs2 `x[rd] = x[rs1] + x[rs2]`

Add. Tipo R, RV32I y RV64I.

Suma el registro x[*rs2*] al registro x[*rs1*] y escribe el resultado en x[*rd*]. Overflow aritmético ignorado.

Formas comprimidas: **c.add** rd, rs2; **c.mv** rd, rs2

31 25	24 20	19 15	14 12	11 7	6 0
0000000	rs2	rs1	000	rd	0110011



Cinquena instrucció

add rd, rs1, rs2 `x[rd] = x[rs1] + x[rs2]`

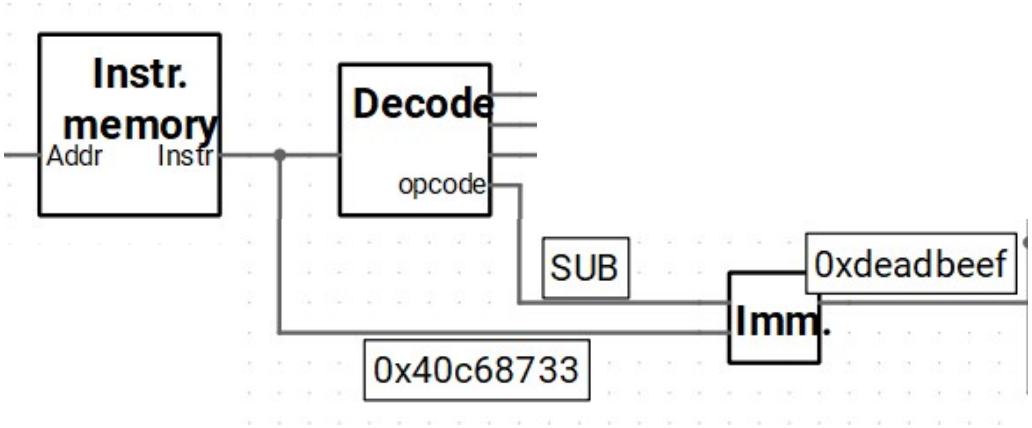
Add. Tipo R, RV32I y RV64I.

Suma el registro x[*rs2*] al registro x[*rs1*] y escribe el resultado en x[*rd*]. Overflow aritmético ignorado.

Formas comprimidas: **c.add** rd, rs2; **c.mv** rd, rs2

31 25	24 20	19 15	14 12	11 7	6 0
0000000	rs2	rs1	000	rd	0110011

- Observeu ara com varia la sortida de l'immediat



Cinquena instrucció

- Exercici:

Indiqueu quines són les entrades que ha de tenir la U.C. i quines són les sortides per executar aquesta instrucció

Sisena instrucció

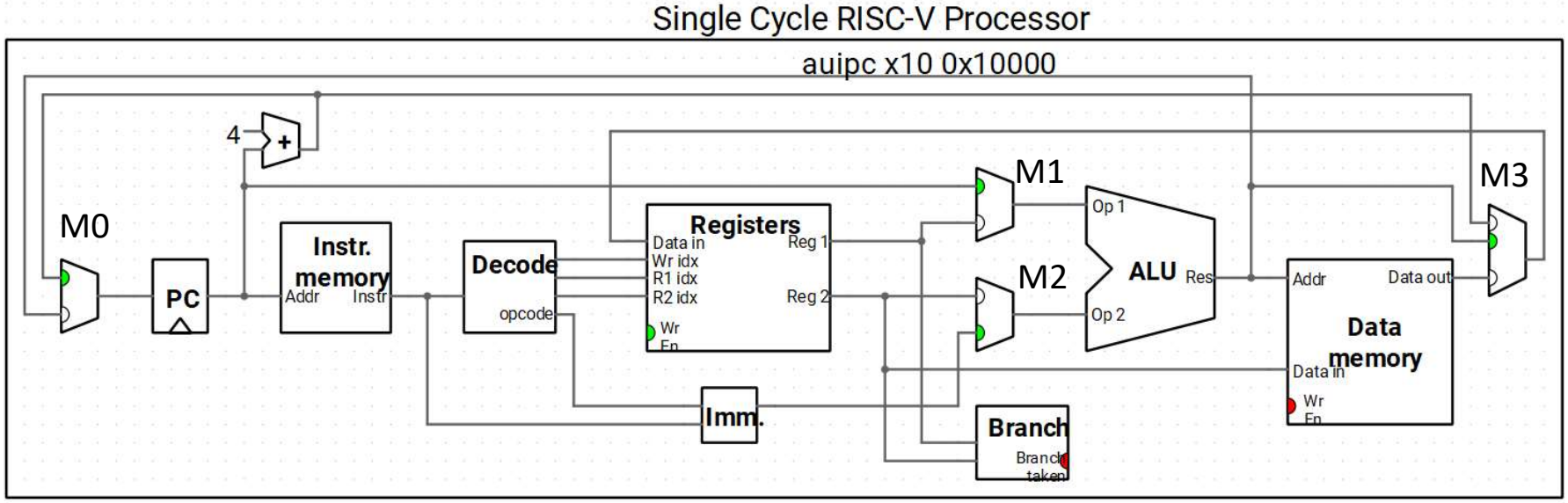
- PC<= PC+4 Enable Write
- X10 <= PC+Imm <<12 bits
- ALU <= ADD
- M0 <=0
- M1 <=1
- M2 <=0
- M3<=01
- DWR<=A
- DLR1 <=x
- DLR2 <=x

auipc rd, immediate `x[rd] = pc + sext(immediate[31:12] << 12)`

Add Upper Immediate to PC. Tipo U, RV32I y RV64I.

Suma el *inmediato* sign-extended de 20 bits, corrido a la izquierda por 12 bits, al *pc*, y escribe el resultado en x[*rd*].

31 — 12	11 — 7	6 — 0
immediate[31:12]	rd	0010111



Sisena instrucció

- Exactament igual que la primera… però que fa la propera instrucción??

Setena instrucció

- PC<= PC+4 Enable Write
- X14 <= [X14] + Imm
- ALU <= ADD
- M0 <=0
- M1 <=0
- M2 <=1
- M3<=01
- DWR<=E
- DLR1 <=E
- DLR2 <=x

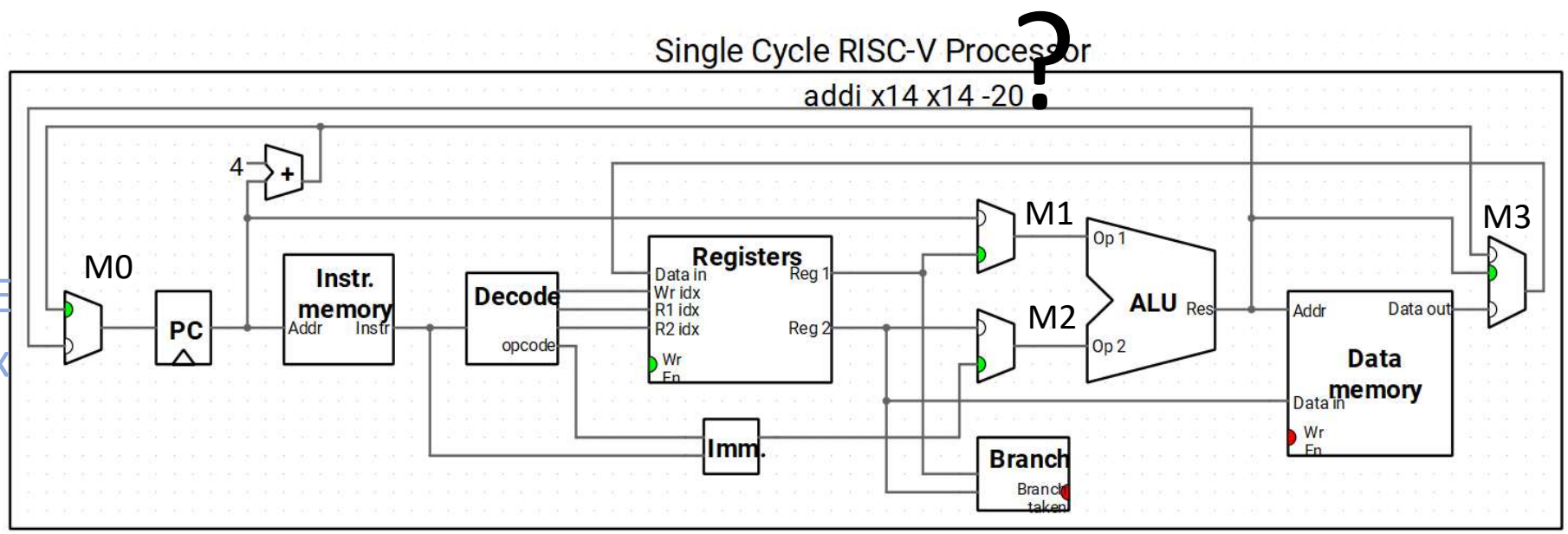
addi rd, rs1, immediate `x[rd] = x[rs1] + sext(immediate)`

Add Immediate. Tipo I, RV32I y RV64I.

Suma el *inmediato* sign-extended al registro x[*rs1*] y escribe el resultado en x[*rd*]. Overflow aritmético ignorado.

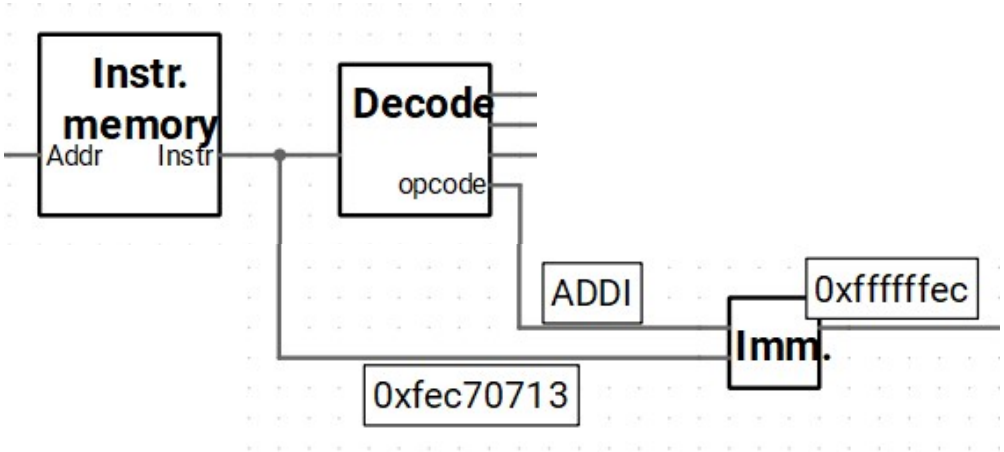
Formas comprimidas: **c.li** rd, imm; **c.addi** rd, imm; **c.addi16sp** imm; **c.addi4spn** rd, imm

31 — 20	19 — 15	14 — 12	11 — 7	6 — 0
immediate[11:0]	rs1	000	rd	0010011



Setena instrucció

- Observeu ara com varia la sortida de l'immediat



- En aquest cas, no hi ha desplaçament del valor, però s'esten al valor del signe => 0xFEC (12 bits) passa a 0xFFFFFFEC

Vuitena instrucció

SW rs2, offset(rs1) `M[x[rs1] + sext(offset)] = x[rs2][31:0]`

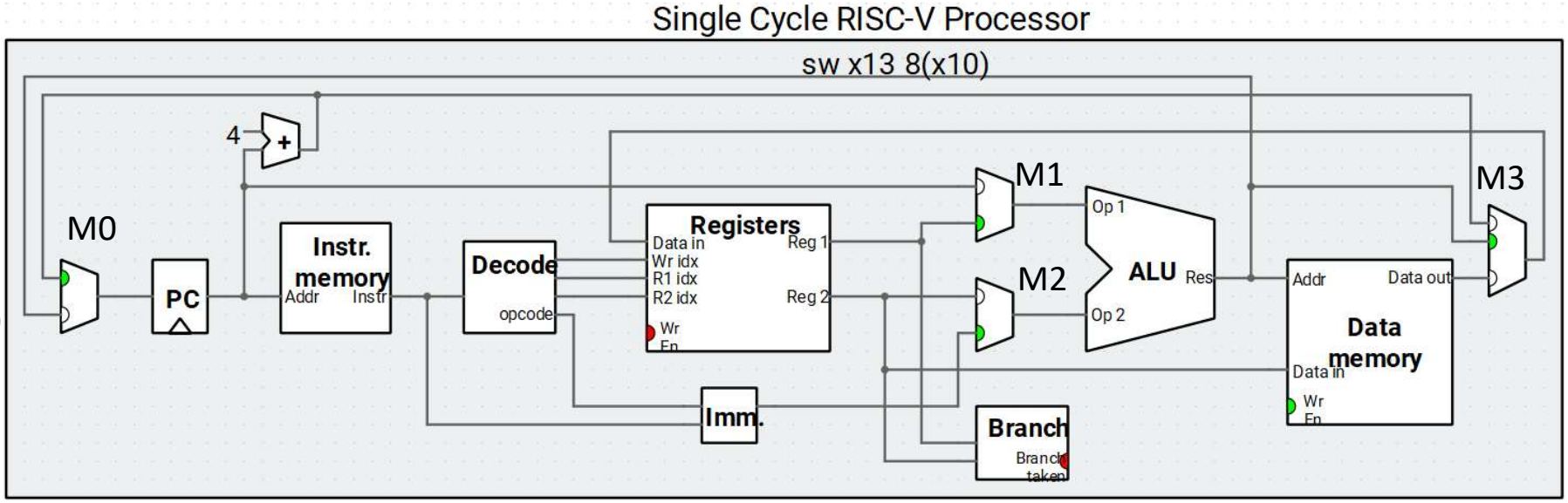
Store Word. Tipo S, RV32I y RV64I.

Almacena los cuatro bytes menos significativos del registro x[*rs2*] a memoria en la dirección x[*rs1*] + *sign-extend*(*offset*).

Formas comprimidas: **c.swsp** rs2, offset; **c.sw** rs2, offset(rs1)

31 25	24 20	19 15	14 12	11 7	6 0
offset[11:5]	rs2	rs1	010	offset[4:0]	0100011

- PC<= PC+4 Enable Write
- @([X10] + Imm)<= X13 Mem Write enabled
- ALU <= ADD
- M0 <=0
- M1 <=0
- M2 <=0
- M3<=01
- DWR<=0
- DLR1 <=A
- DLR2 <=D



Vuitena instrucció

SW rs2, offset(rs1) `M[x[rs1] + sext(offset)] = x[rs2][31:0]`

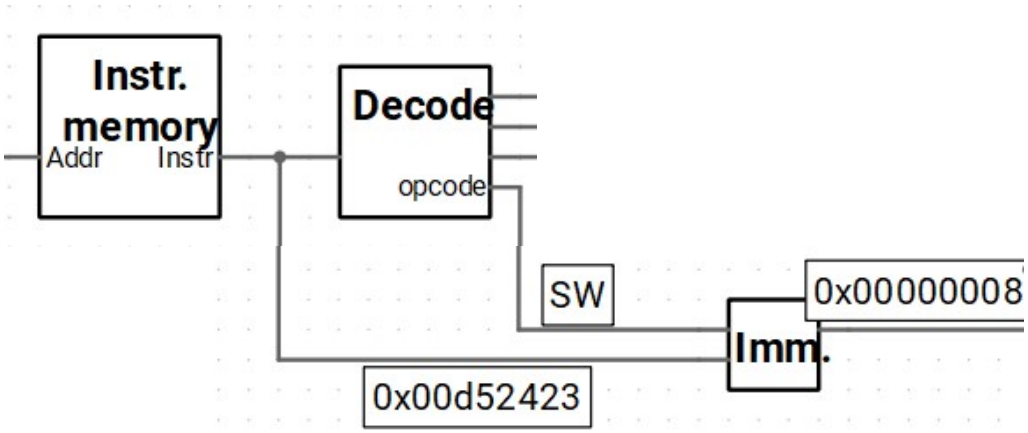
Store Word. Tipo S, RV32I y RV64I.

Almacena los cuatro bytes menos significativos del registro x[*rs2*] a memoria en la dirección x[*rs1*] + *sign-extend*(*offset*).

Formas comprimidas: **c.swsp** rs2, offset; **c.sw** rs2, offset(rs1)

31 25	24 20	19 15	14 12	11 7	6 0
offset[11:5]	rs2	rs1	010	offset[4:0]	0100011

- Com evoluciona ara el càlcul de l'immediat??



Vuitena instrucció

- Exercici:

Indiqueu quines són les entrades que ha de tenir la U.C. i quines són les sortides per executar aquesta instrucció

Exercici final

- Com seria la UC necessària per executar aquest programa?
- Mediteu la resposta