Projet PEP

Processeur ARM Cortex-M

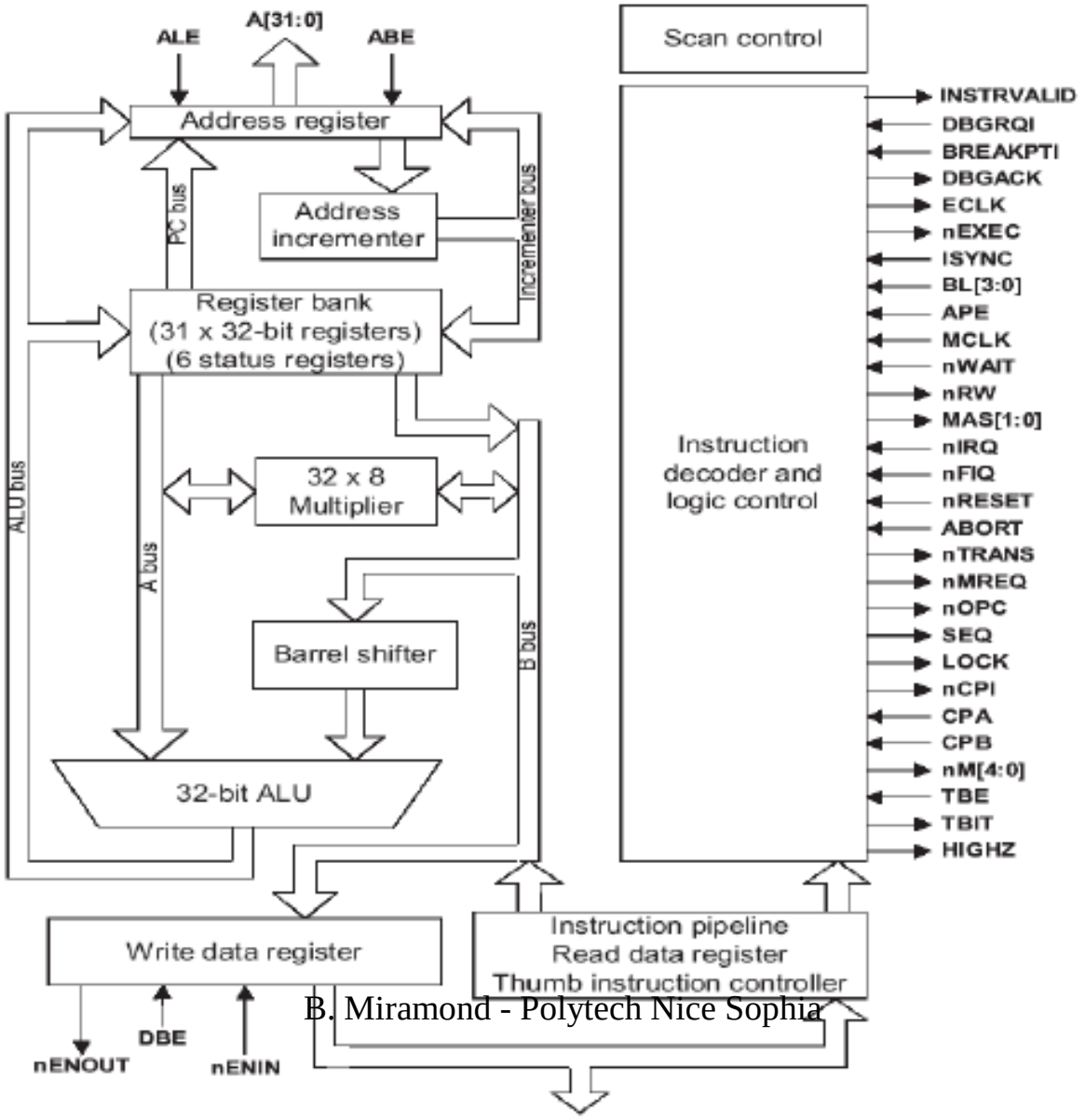
B. Miramond

Polytech Nice

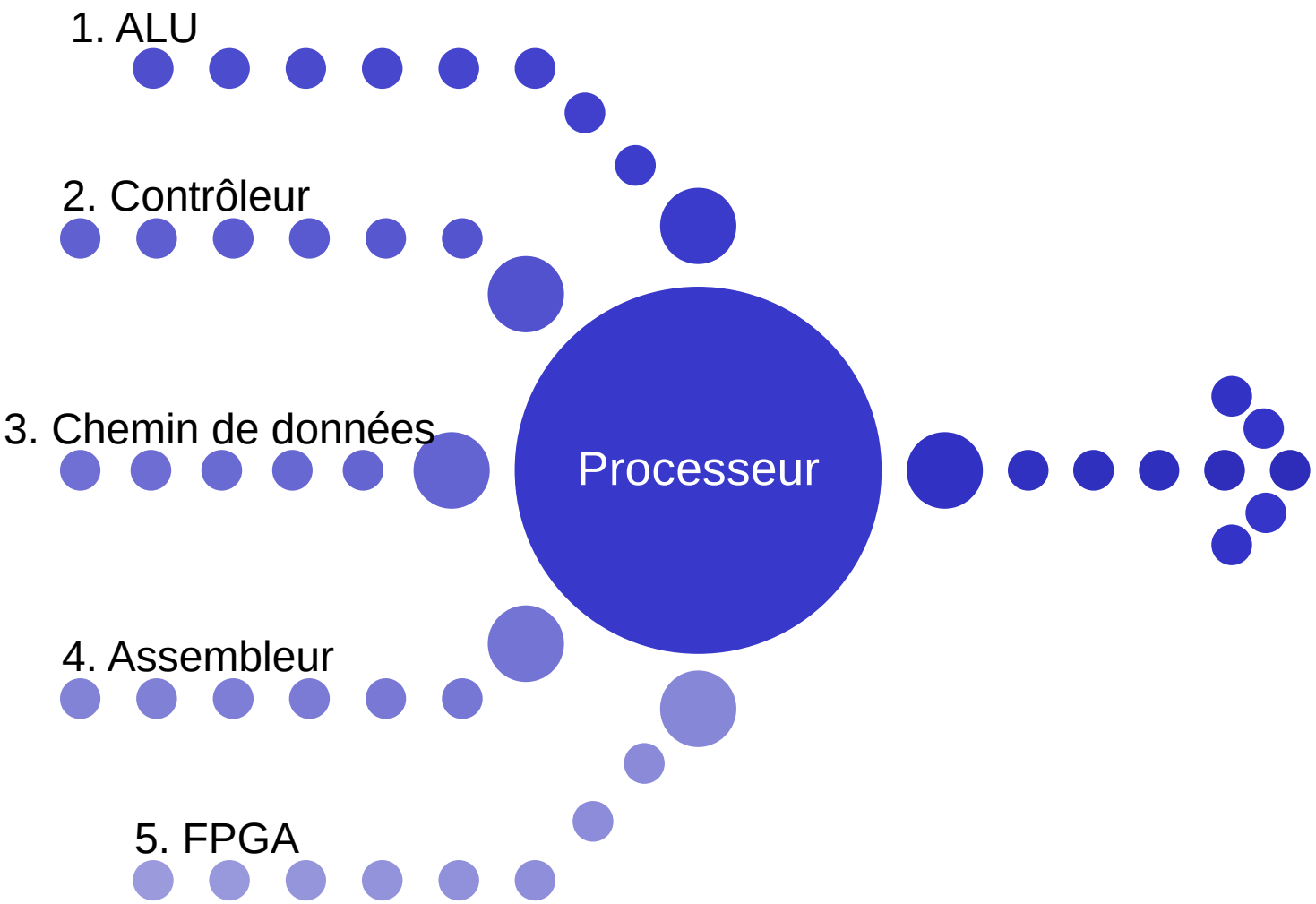
Objectifs du projet

- Concevoir une version allégée du processeur ARM Cortex-M0
  1. Utilisation d'un éditeur de circuits numériques
  2. Conception matérielle
  3. Génération de code binaire depuis l'assembleur
  4. Validation par simulation
  5. Test de déploiement sur FPGA

Architecture du processeur ARM



Répartition des tâches



Rôles de chaque partenaire

ALU (Hardware)	Contrôleur (Hardware)	Chemin de données (Hardware)	Assembleur (Software)	FPGA (Software)
Réaliser les blocs d'opérateurs arithmétiques et logiques	Lecture instruction depuis la mémoire de programme	Mouvement de registre à registre	Parser un fichier assembleur	Tester le déploiement du processeur sur FPGA
Générer les Flags	Décoder les instructions et générer les signaux de commande du chemin de données	Lecture mémoire de données Ecriture en mémoire de données	Générer le fichier binaire à charger dans la mémoire d'instruction de logisim	
	Calcul d'adresse de la prochaine instruction	Envoie d'adresse pour les lectures / écritures	Générer le fichier binaire à charger dans la mémoire de données de logisim	

Projet PEP

7 séances dédiées :

1) Introduction (Décodeur 7-segments)
2) Projet 1 – (ALU, Banc de registres)
3) Projet 2
4) Projet 3
5) Projet 4
6) Intégration, validation
7) Soutenance => Janvier 2016

4 Types d'instructions

a) Shift, add, sub, mov,
   - 7 instructions

b) Data Processing,
   - 16 instructions

c) Load/Store,
   - 2 instructions

d) Branch
   - 1 instruction

Table A5-1 16-bit Thumb instruction encoding

opcode	Instruction or instruction class
00xxxx	*Shift (immediate), add, subtract, move, and compare* on page A5-128
010000	*Data processing* on page A5-129
010001	*Special data instructions and branch and exchange* on page A5-130
01001x	Load from Literal Pool, see *LDR (literal)* on page A7-254
0101xx 011xxx 100xxx	*Load/store single data item* on page A5-131
10100x	Generate PC-relative address, see *ADR* on page A7-197
10101x	Generate SP-relative address, see *ADD (SP plus immediate)* on page A7-193
1011xx	*Miscellaneous 16-bit instructions* on page A5-132
11000x	Store multiple registers, see *STM, STMIA, STMEA* on page A7-422
11001x	Load multiple registers, see *LDM, LDMIA, LDMFD* on page A7-248
1101xx	*Conditional branch, and supervisor call* on page A5-134
11100x	Unconditional Branch, see *B* on page A7-207

Code d'instruction		Catégorie A	Catégorie B	Catégorie C	Catégorie D
00 XX XX	Shift, add, sub…	1			
01 00 00	Data processing		1		
01 10 XX	Load/Store			1	
11 01 XX	Branch				1

Type a)

Table A5-2 16-bit shift (immediate), add, subtract, move and compare encoding

opcode	Instruction	See
000xx	Logical Shift Left[a]	*LSL (immediate)* on page A7-298
001xx	Logical Shift Right	*LSR (immediate)* on page A7-302
010xx	Arithmetic Shift Right	*ASR (immediate)* on page A7-203
01100	Add register	*ADD (register)* on page A7-191
01101	Subtract register	*SUB (register)* on page A7-450
01110	Add 3-bit immediate	*ADD (immediate)* on page A7-189
01111	Subtract 3-bit immediate	*SUB (immediate)* on page A7-448
100xx	Move	*MOV (immediate)* on page A7-312
101xx	Compare	*CMP (immediate)* on page A7-229
110xx	Add 8-bit immediate	*ADD (immediate)* on page A7-189
111xx	Subtract 8-bit immediate	*SUB (immediate)* on page A7-448

Type b)

Table A5-3 16-bit data processing instructions

opcode	Instruction	See
0000	Bitwise AND	*AND (register)* on page A7-201
0001	Exclusive OR	*EOR (register)* on page A7-239
0010	Logical Shift Left	*LSL (register)* on page A7-300
0011	Logical Shift Right	*LSR (register)* on page A7-304
0100	Arithmetic Shift Right	*ASR (register)* on page A7-205
0101	Add with Carry	*ADC (register)* on page A7-187
0110	Subtract with Carry	*SBC (register)* on page A7-380
0111	Rotate Right	*ROR (register)* on page A7-368
1000	Set flags on bitwise AND	*TST (register)* on page A7-466
1001	Reverse Subtract from 0	*RSB (immediate)* on page A7-372
1010	Compare Registers	*CMP (register)* on page A7-231
1011	Compare Negative	*CMN (register)* on page A7-227
1100	Logical OR	*ORR (register)* on page A7-336
1101	Multiply Two Registers	*MUL* on page A7-324
1110	Bit Clear	*BIC (register)* on page A7-213
1111	Bitwise NOT	*MVN (register)* on page A7-328

Type c)

Table A5-5 16-bit Load/store instructions

opA	opB	Instruction	See
0101	000	Store Register	*STR (register)* on page A7-428
0101	001	Store Register Halfword	*STRH (register)* on page A7-444
0101	010	Store Register Byte	*STRB (register)* on page A7-432
0101	011	Load Register Signed Byte	*LDRSB (register)* on page A7-286
0101	100	Load Register	*LDR (register)* on page A7-256
0101	101	Load Register Halfword	*LDRH (register)* on page A7-278
0101	110	Load Register Byte	*LDRB (register)* on page A7-262
0101	111	Load Register Signed Halfword	*LDRSH (register)* on page A7-294
0110	0xx	Store Register	*STR (immediate)* on page A7-426
0110	1xx	Load Register	*LDR (immediate)* on page A7-252
0111	0xx	Store Register Byte	*STRB (immediate)* on page A7-430
0111	1xx	Load Register Byte	*LDRB (immediate)* on page A7-258
1000	0xx	Store Register Halfword	*STRH (immediate)* on page A7-442
1000	1xx	Load Register Halfword	*LDRH (immediate)* on page A7-274
1001	0xx	Store Register SP relative	*STR (immediate)* on page A7-426
1001	1xx	Load Register SP relative	*LDR (immediate)* on page A7-252

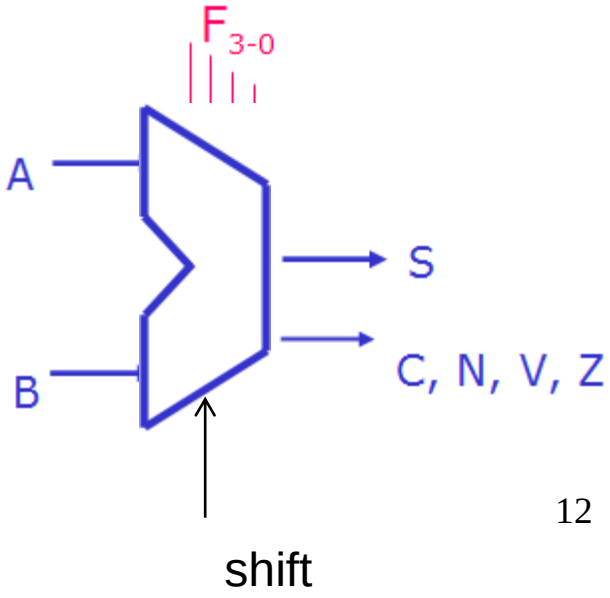
Type d)

Table A5-8 Branch and supervisor call instructions

opcode	Instruction	See
not 111x	Conditional branch	*B* on page A7-207
1110	Permanently UNDEFINED	*UDF* on page A7-471
1111	Supervisor call	*SVC* on page A7-455

Code de commande de l'ALU

F3 F2 F1 F0	Opération	Instructions
0 0 0 0	A and B	AND
0 0 0 1	A xor B	EOR
0 0 1 0	B << shift	LSL
0 0 1 1	B >> shift	LSR
0 1 0 0	B >> shift (arith)	ASR
0 1 0 1	A + B + Cin	ADC
0 1 1 0	A – B + Cin – 1	SBC
0 1 1 1	B >> shift (rot)	ROR
1 0 0 0	A and B	TST
1 0 0 1	0 – B	RSB
1 0 1 0	A – B	CMP
1 0 1 1	A + B	CMN
1 1 0 0	A or B	ORR
1 1 0 1	A * B	MUL
1 1 1 0	A and not B	BIC
1 1 1 1	Not B	MVN



Codes opérations de l'ALU

Sur immédiat principalement :

opcode	Instruction	See
000xx	Logical Shift Left[a]	*LSL (immediate)* on page A7-298
001xx	Logical Shift Right	*LSR (immediate)* on page A7-302
010xx	Arithmetic Shift Right	*ASR (immediate)* on page A7-203
01100	Add register	*ADD (register)* on page A7-191
01101	Subtract register	*SUB (register)* on page A7-450
01110	Add 3-bit immediate	*ADD (immediate)* on page A7-189
01111	Subtract 3-bit immediate	*SUB (immediate)* on page A7-448
100xx	Move	*MOV (immediate)* on page A7-312
101xx	Compare	*CMP (immediate)* on page A7-229
110xx	Add 8-bit immediate	*ADD (immediate)* on page A7-189
111xx	Subtract 8-bit immediate	*SUB (immediate)* on page A7-448

Sur Registres principalement :

opcode	Instruction	See
0000	Bitwise AND	*AND (register)* on page A7-201
0001	Exclusive OR	*EOR (register)* on page A7-239
0010	Logical Shift Left	*LSL (register)* on page A7-300
0011	Logical Shift Right	*LSR (register)* on page A7-304
0100	Arithmetic Shift Right	*ASR (register)* on page A7-205
0101	Add with Carry	*ADC (register)* on page A7-187
0110	Subtract with Carry	*SBC (register)* on page A7-380
0111	Rotate Right	*ROR (register)* on page A7-368
1000	Set flags on bitwise AND	*TST (register)* on page A7-466
1001	Reverse Subtract from 0	*RSB (immediate)* on page A7-372
1010	Compare Registers	*CMP (register)* on page A7-231
1011	Compare Negative	*CMN (register)* on page A7-227
1100	Logical OR	*ORR (register)* on page A7-336
1101	Multiply Two Registers	*MUL* on page A7-324
1110	Bit Clear	*BIC (register)* on page A7-213
1111	Bitwise NOT	*MVN (register)* on page A7-328

Comparaison des instructions travaillant sur registre et sur immédiat

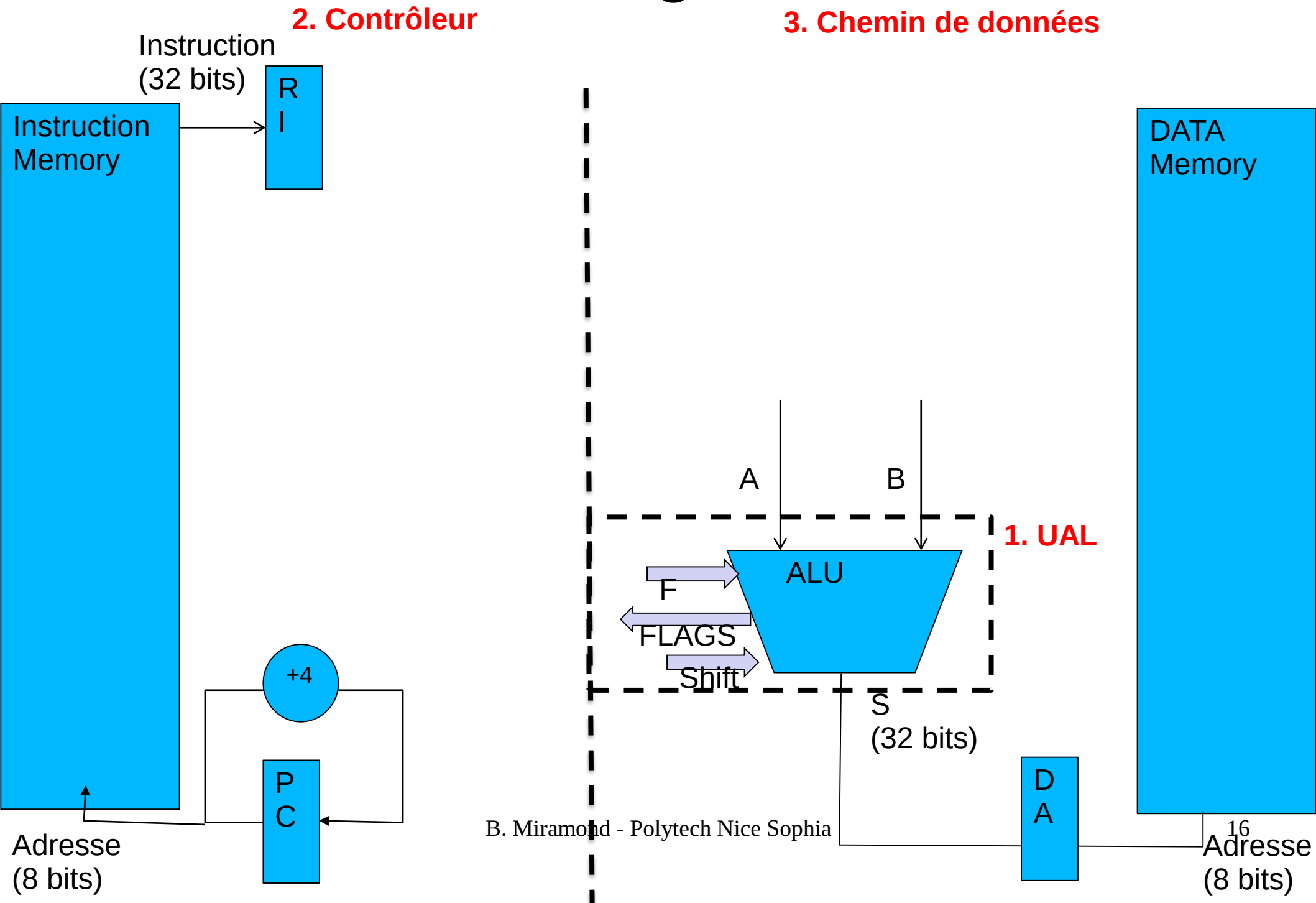
Opcod	Inst	Opcod	Inst
000xx	LSL I	0000	AND
001xx	LSR I	0001	EOR
010xx	ASR I	0010	LSL
01100	ADD R	0011	LSR
01101	SUB R	0100	ASR
01110	ADD I3	0101	ADC
01111	SUB I3	0110	SBC
100xx	MOV I	0111	ROR
101xx	CMP I	1000	TST
110xx	ADD I8	1001	RSB
111xx	SUB I8	1010	CMP
		1011	CMN
		1100	ORR
		1101	MUL
		1110	BIC
		1111	MVN

Répartition des rôles

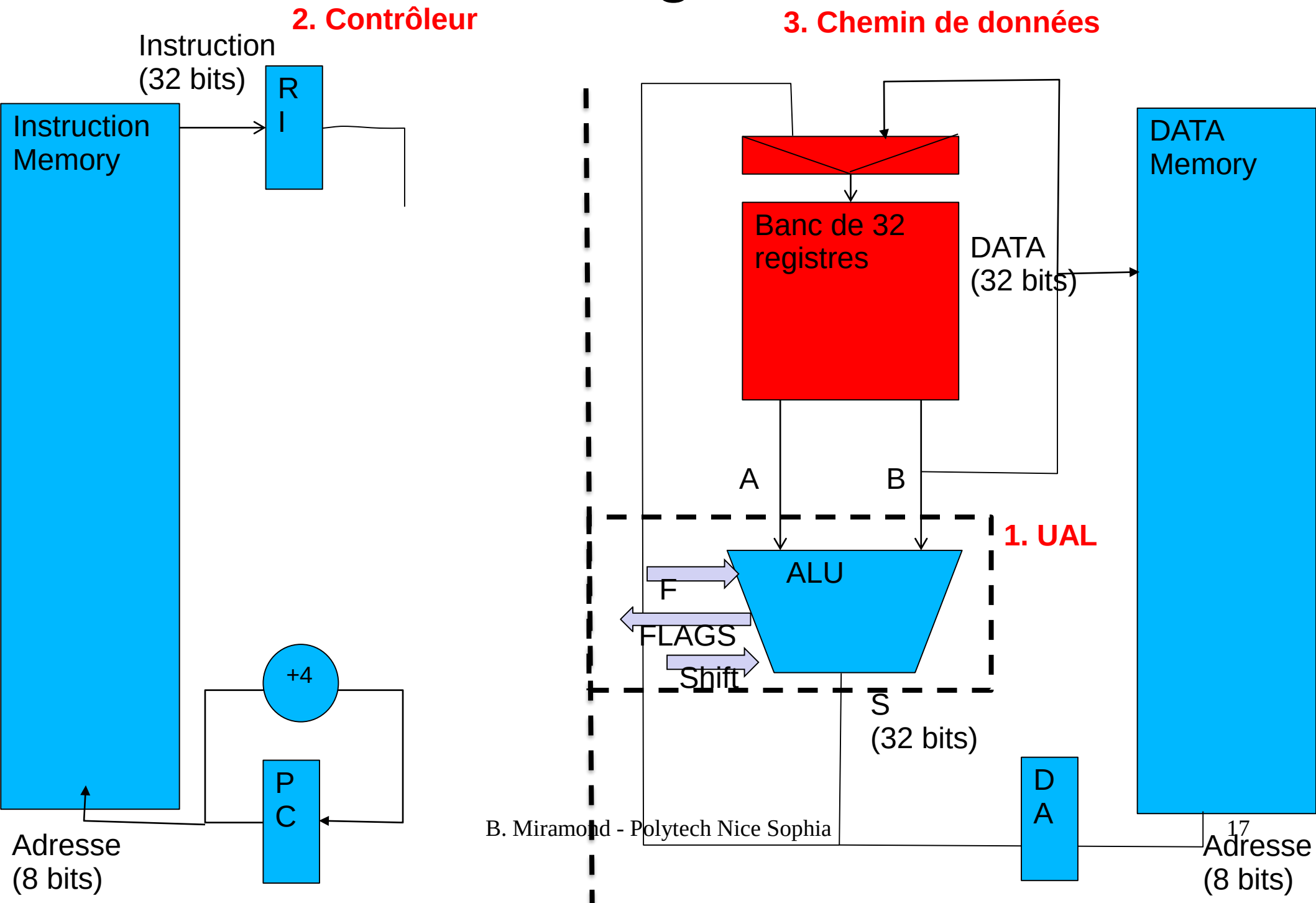
Chaque groupe dispose de 4 séances :

- G1 – Réalisation de l'UAL
- G2 - Contrôleur-MémoireI
- G2 – Registres-MémoireD
- G3 - Assembleur
  - Remplissage mémoire
- G4 - FPGA
  - Déploiement sur FPGA
  - Environnement de test
- Ecriture de jeux de test

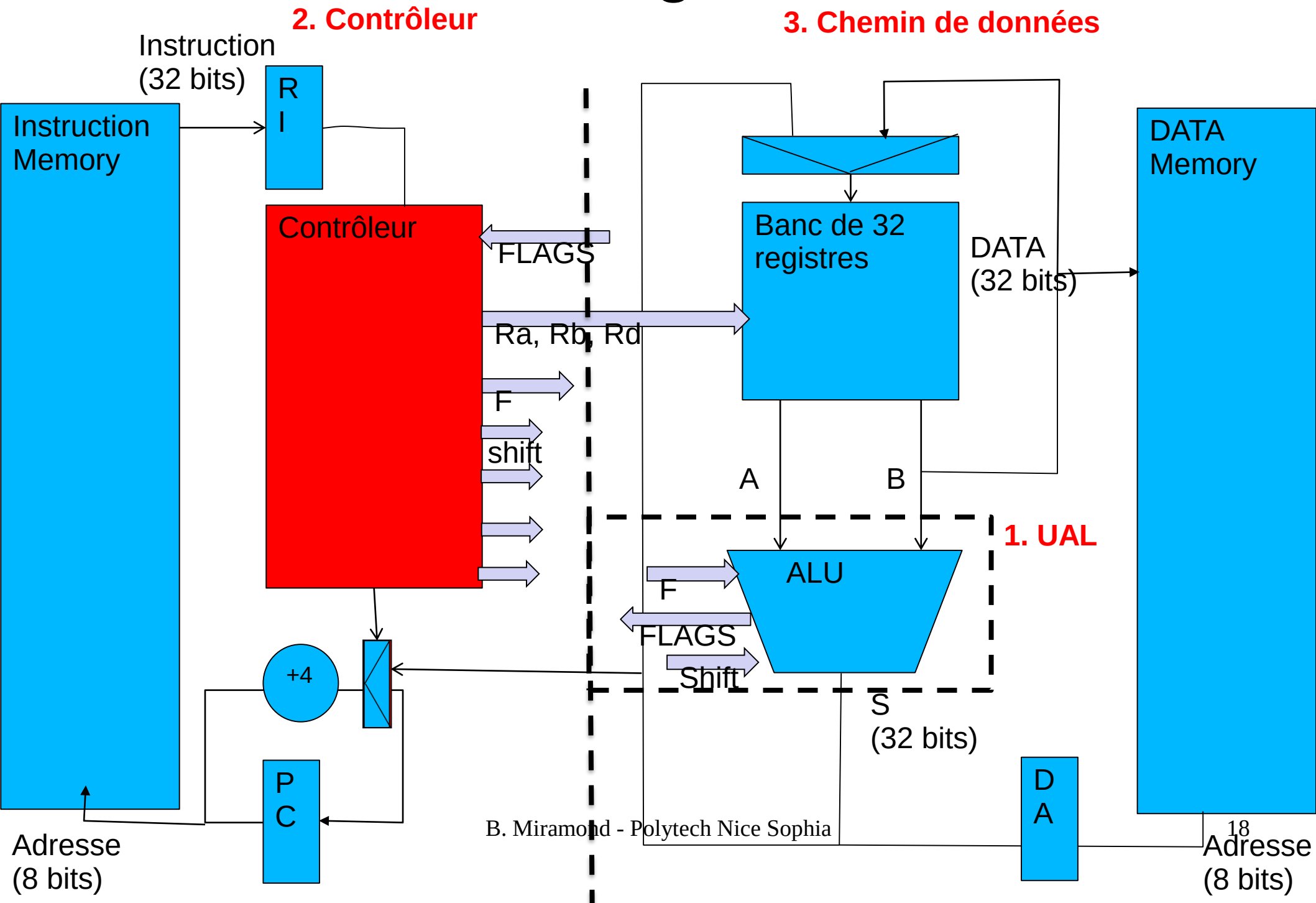
Architecture générale



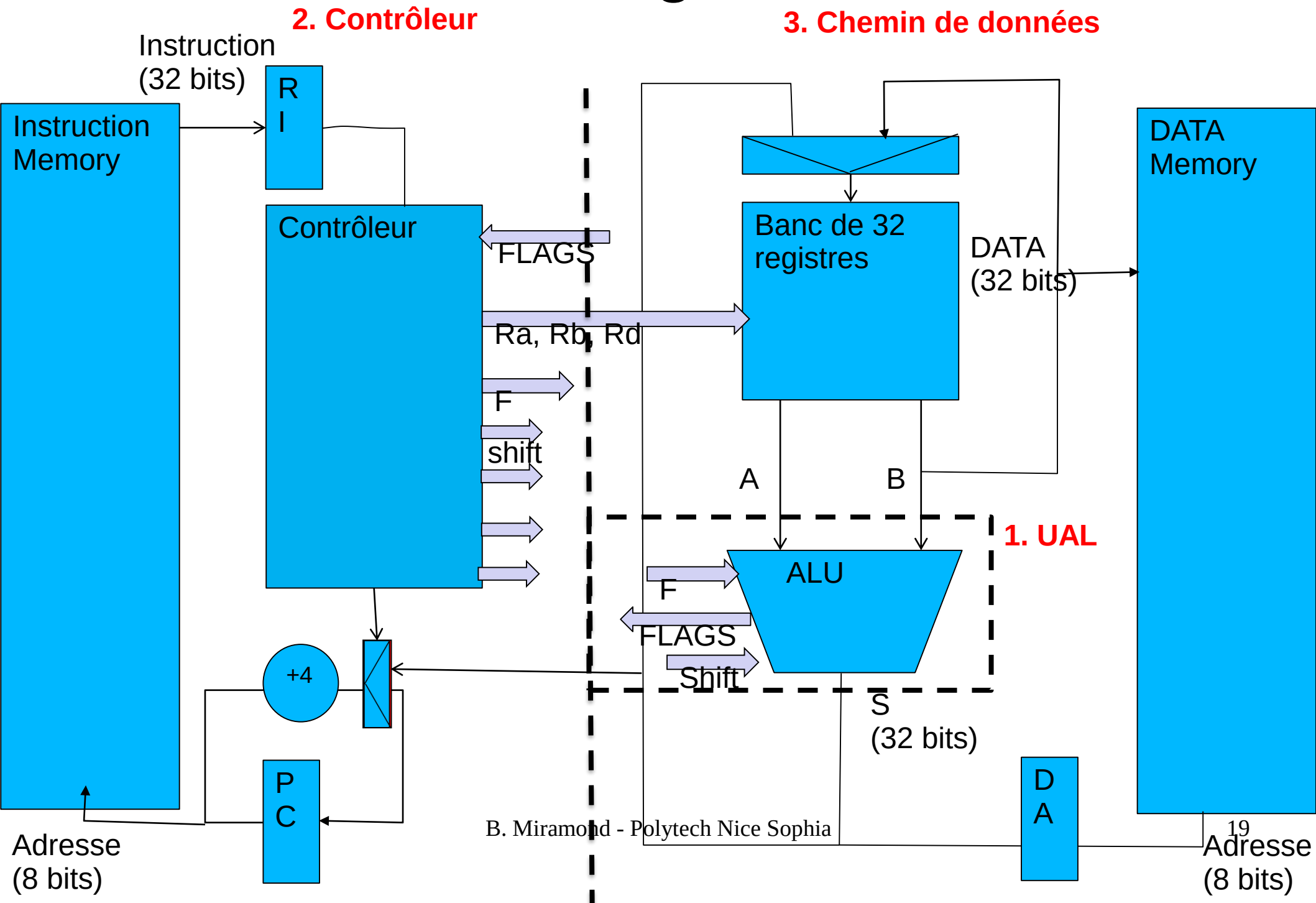
Architecture générale



Architecture générale



Architecture générale



Groupes logiciels

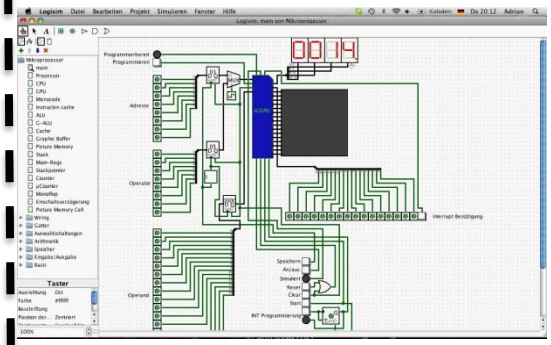
4. Assembleur

```
pgcd:
while:  cmp R1,R2
                bne do
                b endwhile
do:
if:             cmp R1,R2
                bhi then
                b else
then:   sub R1,R1,R2
                b endif
else:   sub R2,R2,R1
endif:  b while
endwhile:
                mov pc,R14
```

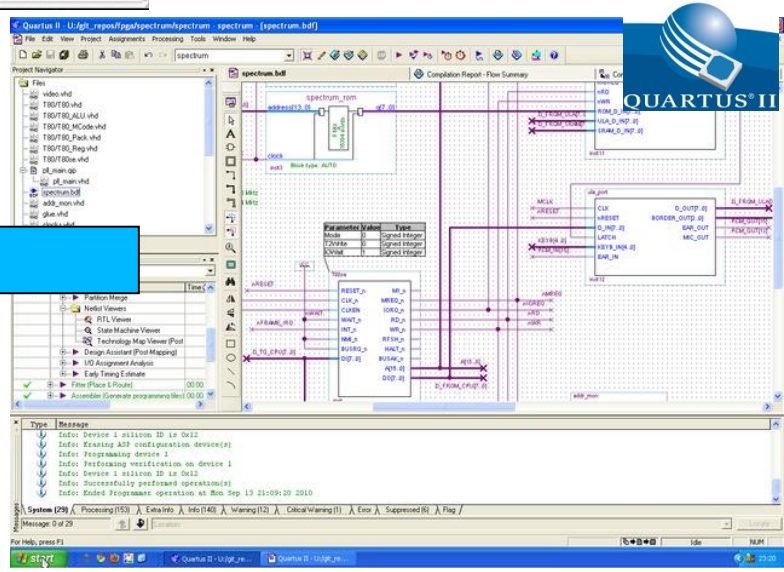
0x2345 4569
0x0124 4828
0x2839 4018
0x3282 0294
…

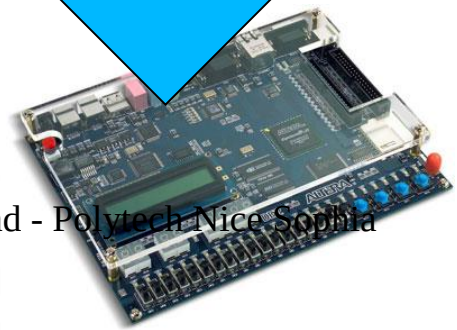
Fichier binaire d'initialisation des mémoires pour Logisim
+
Désassembleur

5. FPGA



Déploiement et Tests





Carte embarquée FPGA DE2

Groupes logiciels

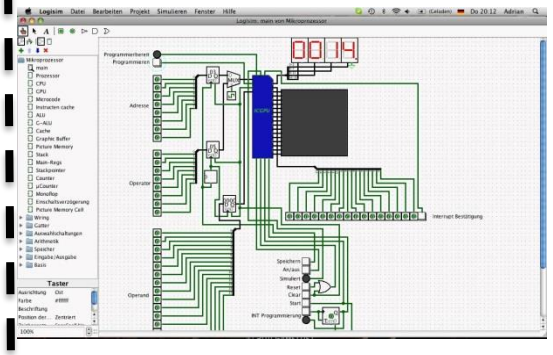
4. Assembleur

```
pgcd:
while:  cmp R1,R2
                bne do
                b endwhile
do:
if:             cmp R1,R2
                bhi then
                b else
then:   sub R1,R1,R2
                b endif
else:   sub R2,R2,R1
endif:  b while
endwhile:
                mov pc,R14
```

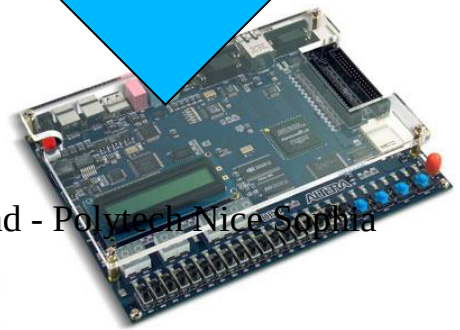
0x2345 4569
0x0124 4828
0x2839 4018
0x3282 0294
…

Fichier binaire d'initialisation des mémoires pour Logisim
+
Désassembleur

5. CAO



- Installer Quartus d'Altera
- Récupérer l'archive logisim : https://github.com/reds-heig/logisim-evolution
- Tester quelques composants sur carte FPGA DE0 Nano
- Modifier le code de logisim pour ajouter une nouvelle carte : DE2
- Vérifier la synthèse des mémoires de code et de données



Carte embarquée FPGA DE2

Constituer les groupes !

4 personnes par groupe, dont 1 chef d'équipe.

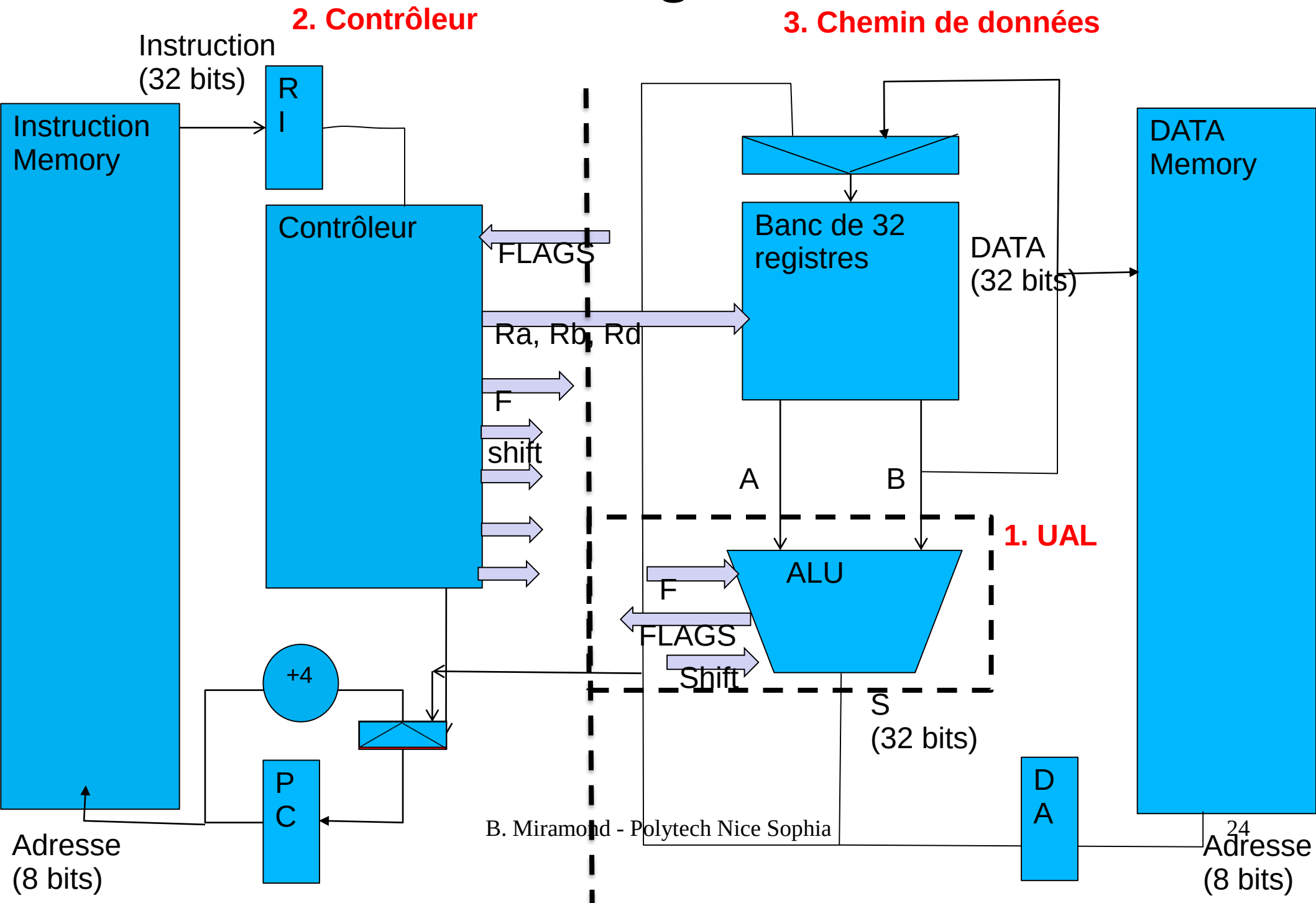
Ré-organisation des efforts du groupe en fonction de l'avancement.

Soutenance de 20 minutes par groupe :

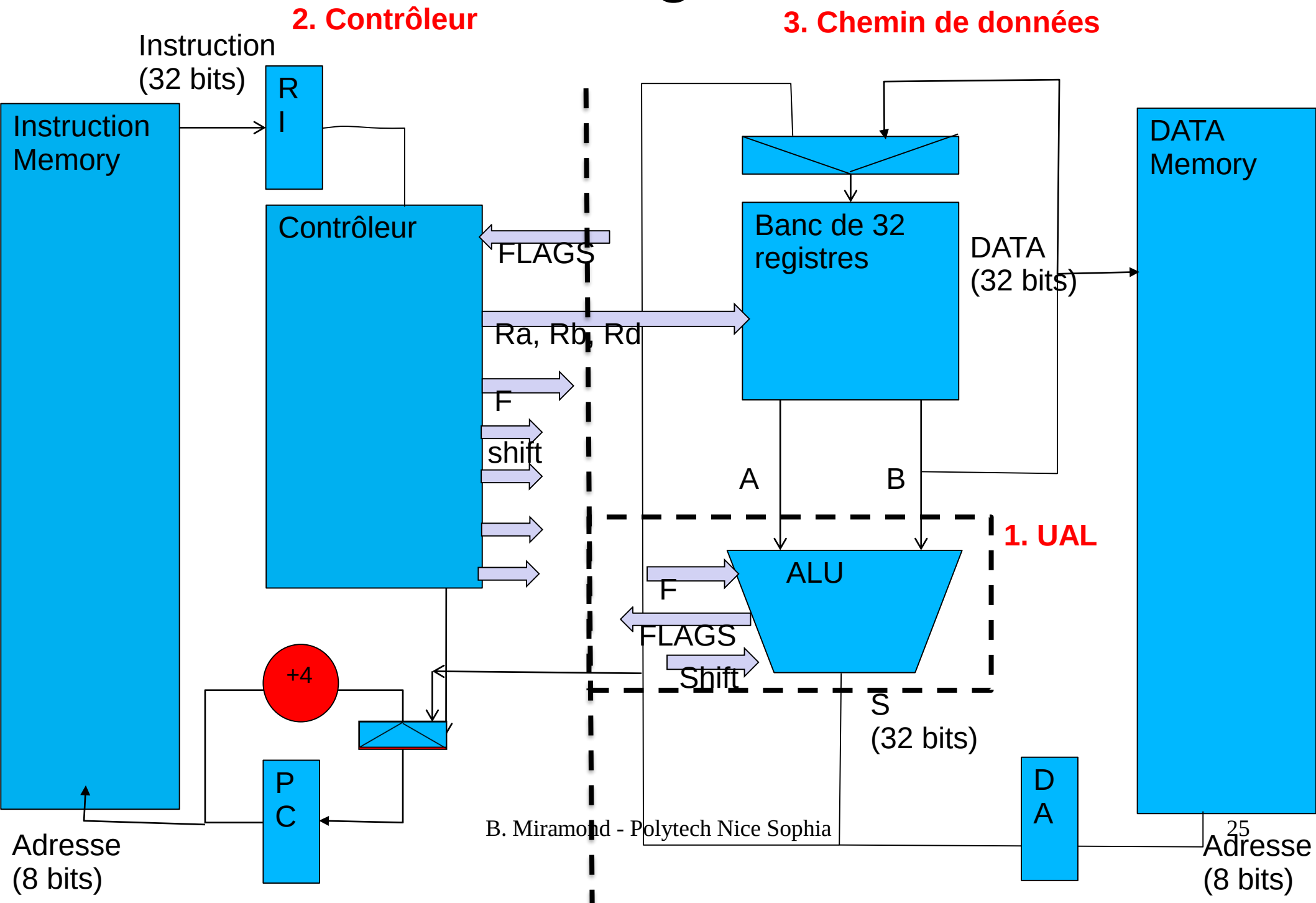
- 20 planches maximum
- Présentation des 5 parties du projet
- Présentation des résultats
- Répétitions obligatoires

Synchronisation

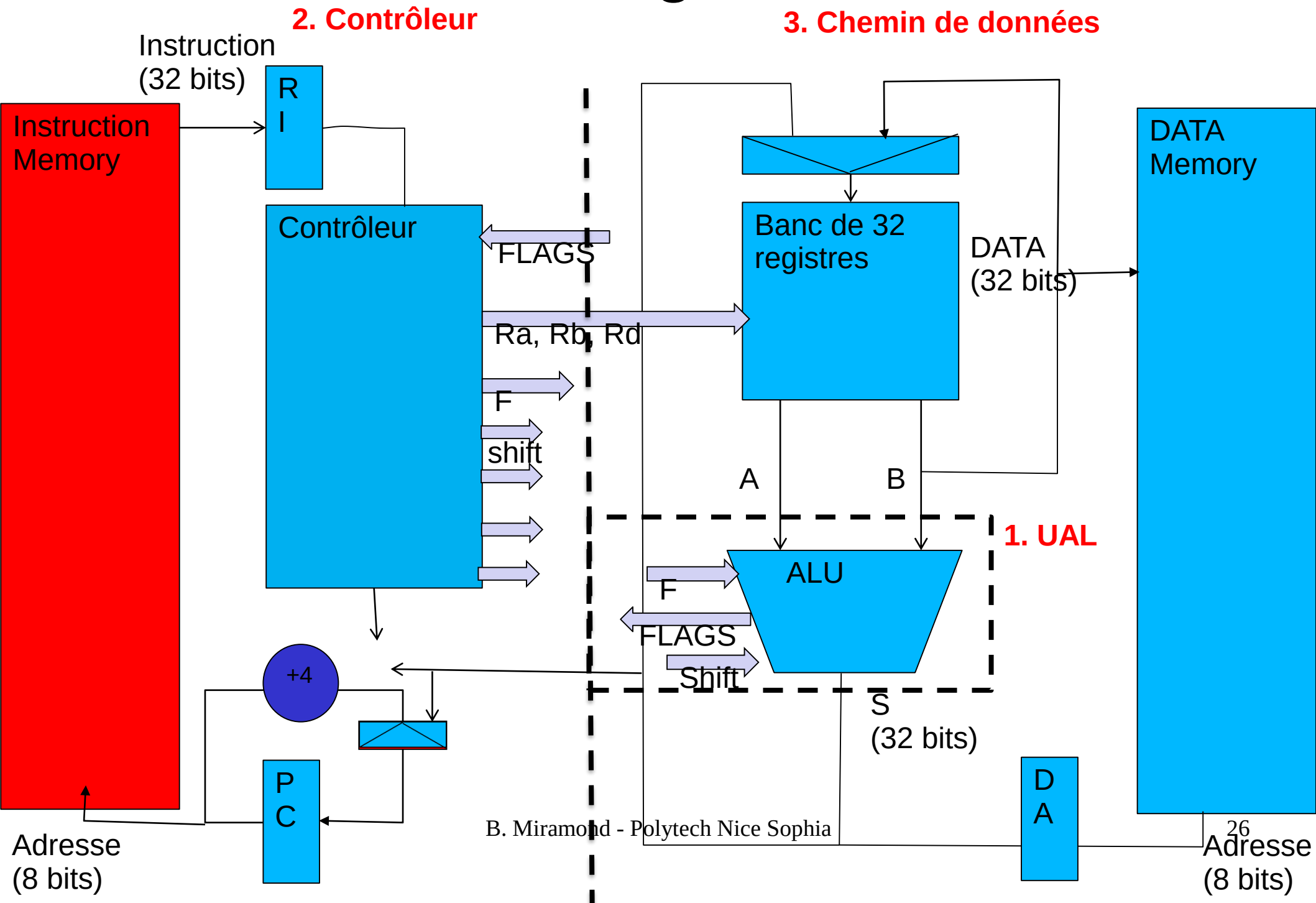
Architecture générale



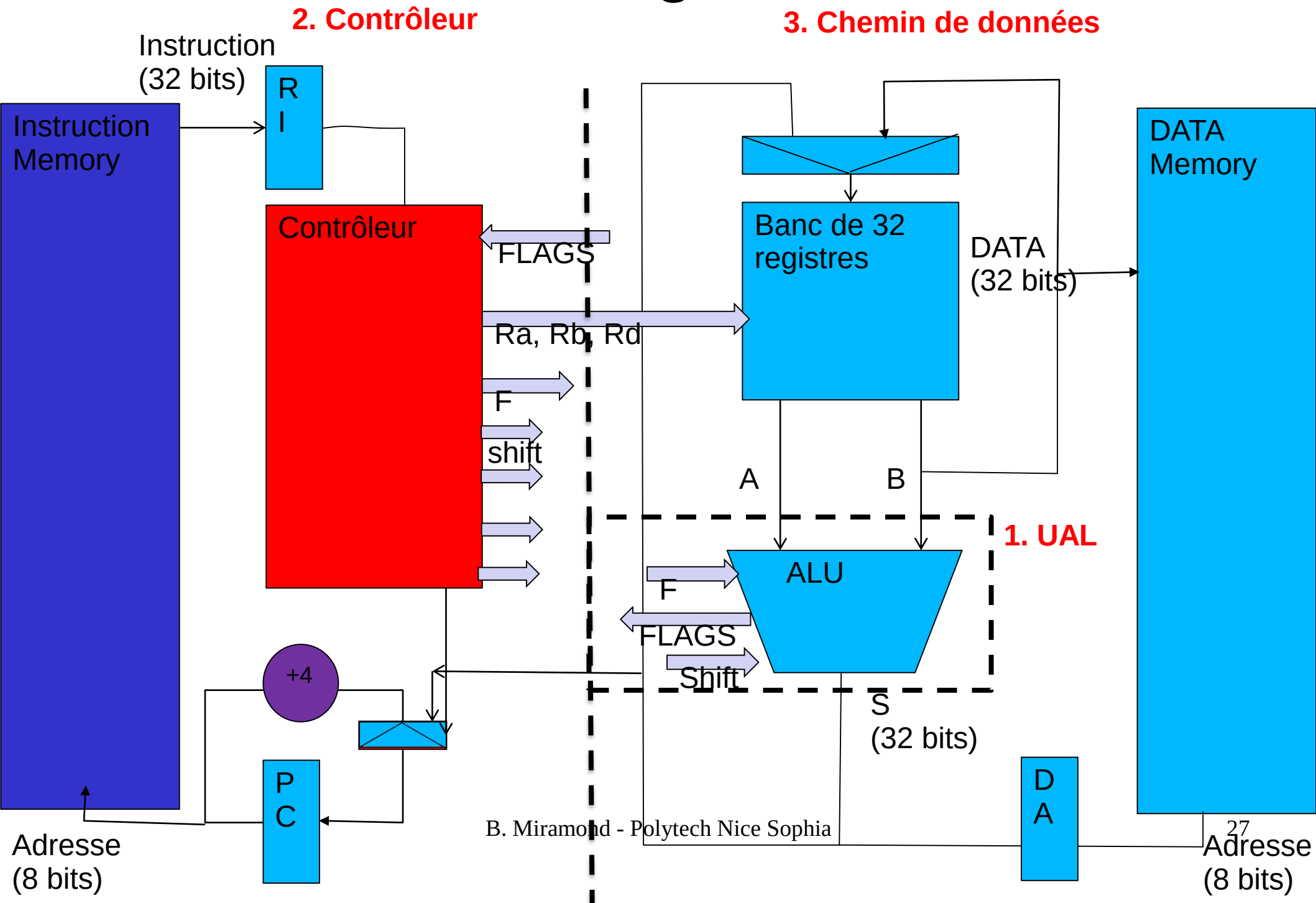
Architecture générale



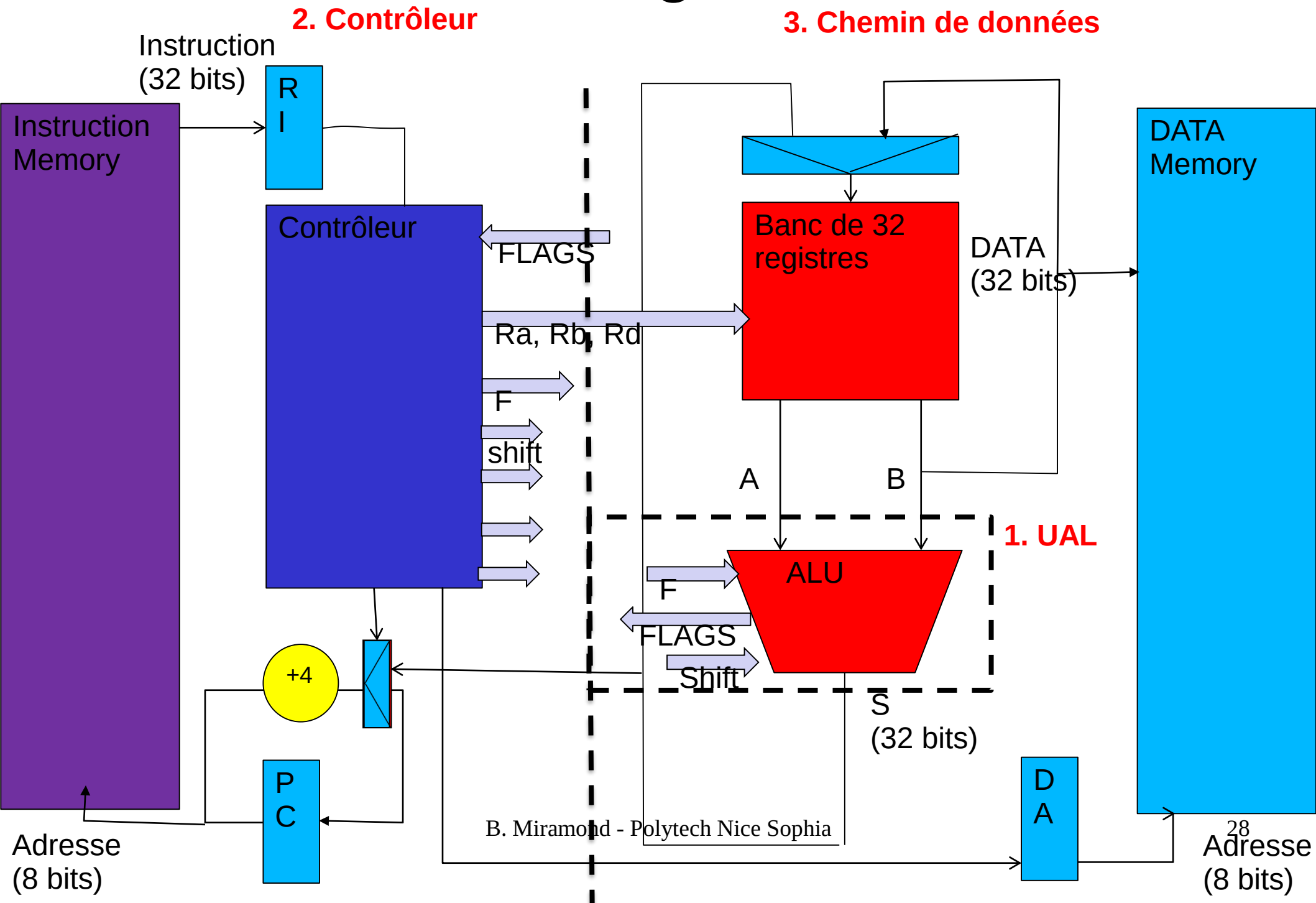
Architecture générale



Architecture générale



Architecture générale



Architecture générale