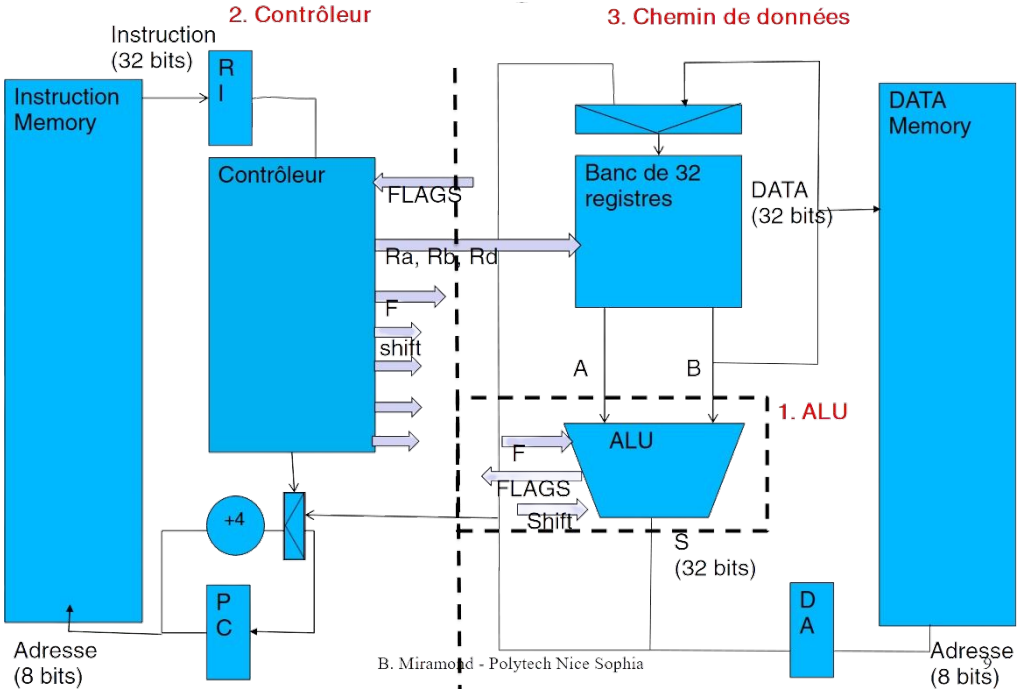
Projet PARM

Hajar El Gholabzouri
Gwendolyne Bouchard
Camille Antonios
Amélie Muller

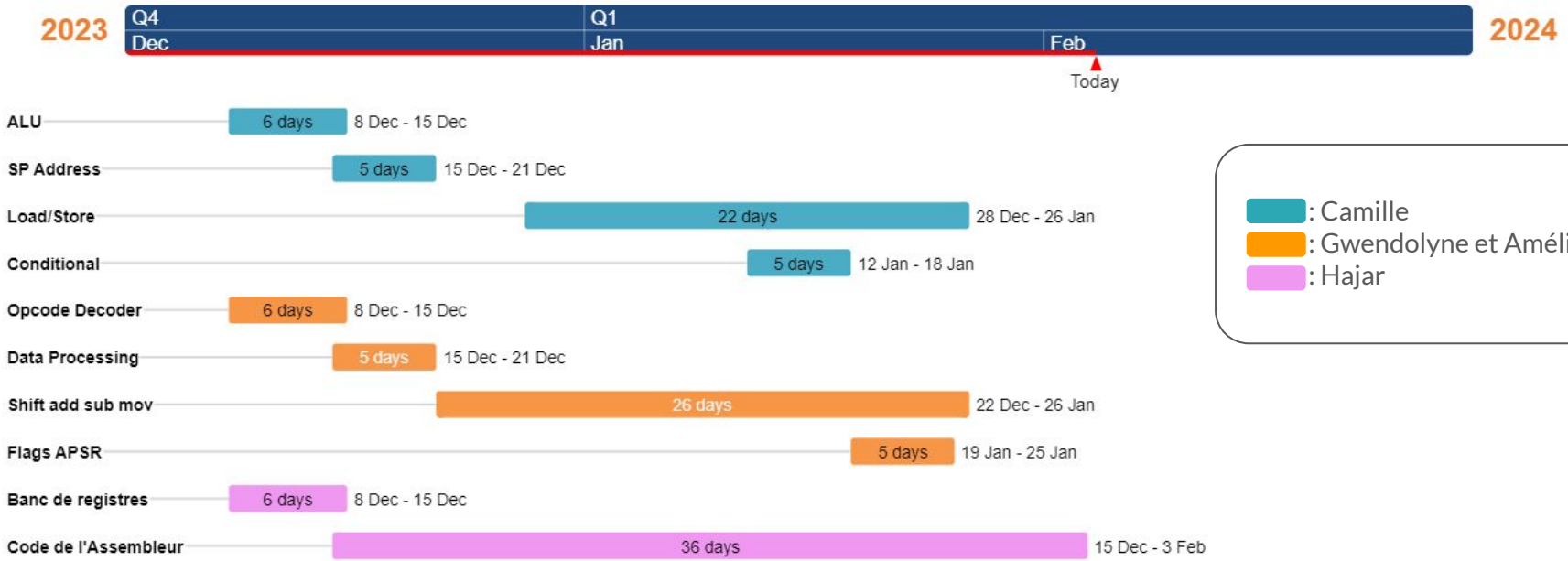
SI3 Février 2024

Architecture générale

- **But** : Reconstruire un microprocesseur ARM capable d'exécuter une liste d'instruction
- Fonctionnement du Processeur



Répartition du travail



Difficultés rencontrées

- Compréhension globale du projet
- Manque de connaissances globale en Électronique au début du projet
- Code de l'assembleur : écriture du parseur fastidieuse
- Debuggage
- Modélisation de certains composants :
  - Mise en place du Load/Store
  - Mise en place du Shift, add, sub, mov

Qualité du projet

Bonne qualité	Mauvaise qualité
• Projet complet : toutes les instructions de la doc sont gérées • Tous les tests unitaires passent • Tous les tests d'intégration passent • Tableau Excel regroupant toutes les instructions en binaire très clair	• Doute sur l'efficacité de l'architecture des branchements (peu de connaissances en électronique avant le projet) • Code de l'assembleur long, complexe et peu lisible • les tests assembleur avec code c ne fonctionne pas tous

Tests

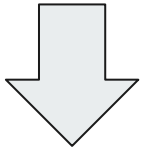
Codes ASM	test passe	non testé
Conditional	1/1	0/1
DP_1_4	1/1	0/1
DP_5_10	1/1	0/1
DP_11_12	1/1	0/1
DP_13_16	1/1	0/1
Load_store	1/1	0/1
SP	1/1	0/1
SASM_1_4	1/1	0/1
SASM_5_8	1/1	0/1
taux de couverture	100%	0%

Codes C	test asm passe	test logisim passe	non testé
calckeyb	0/1	1/1	1/1
calculator	0/1	1/1	1/1
simple_add	1/1	1/1	0/1
test fp	0/1	1/1	1/1
tty	1/1	1/1	0/1
my own test	0/0	0/0	0/0
taux de couverture	40%	100%	60%

Démonstration Assembleur

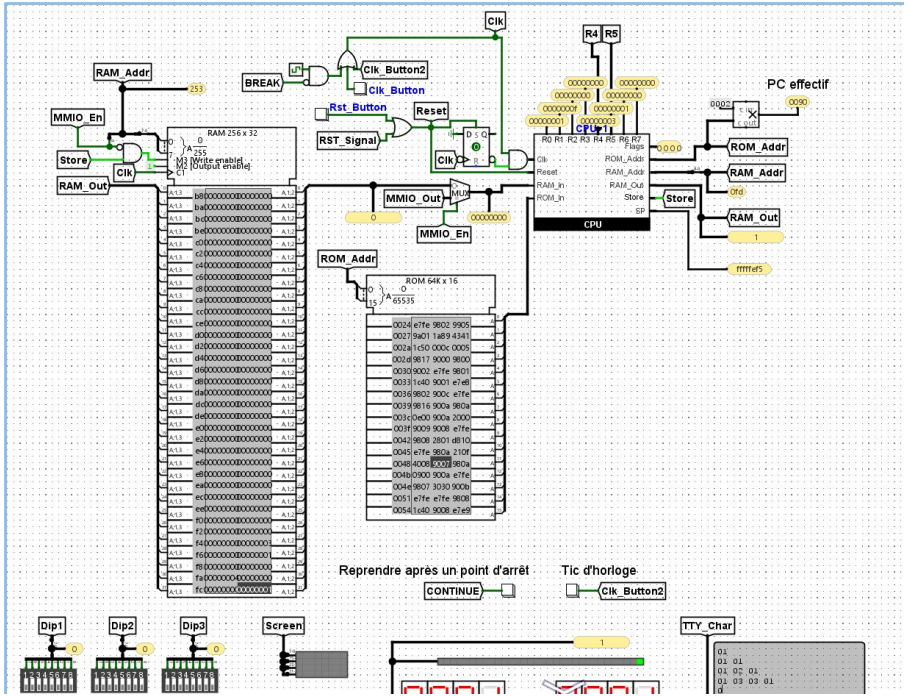
Rôle du code assembleur :
Transformer des instructions assembleur en hexadécimal

```
run:
        .fnstart
        .pad    #96
        sub     sp, #96
        @APP
        sub     sp, #508
        @NO_APP
        @APP
        sub     sp, #452
        @NO_APP
        movs    r0, #1
        str     r0, [sp, #8]
        movs    r0, #2
        str     r0, [sp, #4]
        ldr     r0, [sp, #8]
        ldr     r1, [sp, #4]
        adds    r0, r0, r1
        str     r0, [sp]
        ldr     r0, [sp]
        str     r0, [sp, #36]
        b       .LBB0_1
.LBB0_1:
        b       .LBB0_2
.LBB0_2:
        b       .LBB0_2
.Lfunc_end0:
        .size   run, .Lfunc_end0-run
        .cantunwind
        .fnend
```



```
b098 b0ff b0f1 2001 9002 2002 9001 9802 9901 1840 9000 9800 9009 e7fe e7fe e7fd
```

Démonstration Processeur



Merci pour votre écoute