Tests unitaires :

Nom test	Validé	Description
Test shift add sub mov	OK	test unitaire qui couvre toutes les possibilité pour le composant shift add sub mov
Test conditional	OK	test unitaire qui couvre toutes les possibilité pour le composant conditionnal
Test sp address	OK	test unitaire qui couvre toutes les possibilité pour le composant sp address
Test op code	OK	test unitaire qui couvre toutes les possibilité pour le composant op code
Test data processing	OK	test unitaire qui couvre toutes les possibilité pour le composant data processing
Test load store	OK	test unitaire qui couvre toutes les possibilité pour le composant load store
Test alu	OK	test unitaire qui couvre toutes les possibilité pour le composant alu

Tests assembler :

Nom test	Validé	Description
Test conditional	OK	branch.s teste les branches conditionnelles en boucle. Il initialise r0 = 0, r1 = 1, et r2 = 20, puis compare r0 et r1 pour inverser r2 si r0 < r1. Ensuite, il vérifie si r2 < r1 et, si vrai, assigne 50 à r0 et recommence la boucle. La somme finale r3 = r0 + r2 doit être 70 ou 46, validant le comportement des sauts conditionnels.
Test data processing	OK	1-4_instructions.s teste les opérations de traitement de données (ANDS, EORS, LSLS, LSRS). 11-12_instructions.s teste les instructions de comparaison (CMP, CMN) et leur effet sur les drapeaux du registre d'état (NZCV). 13-16_instructions.s teste les opérations logiques et arithmétiques tel que OU logique, ET logique, etc. 5-10_instructions.s teste les opérations arithmétiques, logiques et de rotation suivantes tel que le décalage vers la droite ...
Test load store	OK	load_store.s teste le stockage (STR) et le chargement (LDR) avec le pointeur de pile (sp). Il initialise r0 = 170 (0xAA), r1 = 255 (0xFF), et ajuste sp. r0 est stocké à sp + 4 et r1 à sp + 0, puis sp est décrémenté. Ensuite, r2 recharge la valeur depuis sp + 4, confirmant la bonne gestion mémoire. Il valide l'implémentation des opérations sur sp et l'accès en RAM.
Test miscellaneous	OK	sp.s teste la modification du pointeur de pile (sp). Il commence par ajouter 16 à sp, puis soustrait 4, modifiant ainsi sa valeur. Lors du premier état haut, sp est fixé à 4, et lors du second, il est ajusté à 3. Ce test vérifie la gestion des opérations arithmétiques sur le registre sp, essentiel pour la gestion de la pile en assembleur.
Test shift add sub mov	OK	1-4_instructions.s teste les instructions de décalage, addition et soustraction (LSLS, LSRS, SUBS, ASRS, ADDS). 5-8_instructions.s teste les opérations arithmétiques de base tel que subb add mov ...

Tests c :

Nom test	Validé	Description
calckeyb.c	OK	calckeyb.c implémente une calculatrice interactive prenant deux entiers en entrée (a et b). L'utilisateur choisit une opération (+ - * / % & | ^) via READKEY(), et le programme exécute le calcul correspondant. Les résultats sont affichés avec PRINTRES_SIGN(), et une attente (WAITKEY()) permet de visualiser le résultat avant réinitialisation (RESET()).
calculator.c	OK	calculator.c implémente une calculatrice matérielle utilisant des entrées (OPa, OPb, CMD) issues de commutateurs (DIP1, DIP2, DIP3). Selon la valeur de CMD, il effectue une addition, une soustraction, une multiplication ou un décalage à gauche (LSL). Le résultat est stocké dans RES et mis à jour en boucle.
simple_add.c	OK	simple_add.c vérifie le bon fonctionnement d'une addition simple en initialisant deux variables (a = 1, b = 2), puis en stockant leur somme dans c. Le résultat est assigné à RES pour validation.
testfp.c	OK	testfp.c vérifie les calculs en virgule fixe (fixed_t). Il initialise deux valeurs (3.5 et 7.5), effectue leur multiplication (MULTFP), puis affiche le résultat. Ensuite, il calcule la racine carrée (SQRTFP) de ce produit et l'affiche. Enfin, il divise cette valeur par 1.5 (DIVFP) et affiche le résultat.
tty.c	OK	tty.c vérifie l'affichage de texte en imprimant "Projet PARM" via PUTCHAR().