Documentation des Fonctions de l'Assembleur M2 MIC

Adya Sarr et Yazici Servan

Octobre 2025

Projet TP n°1 : Informatique Embarquée — Codage d'un Jeu d'Instruction

1 Fonctions Utilitaires et de Parsing

Ces fonctions sont utilisées pour préparer et analyser les éléments de base des lignes d'assembleur.

Fonction	Rôle et Détail
`void netoyage_file(char *str)`	**Nettoyage de Ligne.** Élimine les commentaires (#), supprime les espaces et tabulations inutiles en début et fin de chaîne, et convertit le texte en minuscules (`tolower`). Elle assure une analyse uniforme des instructions.
`int get_opcode(const char *op_name)`	**Récupération d'Opcode.** Consulte la table de correspondance `map_op` pour trouver et retourner la valeur binaire (sur 8 bits) associée au nom de l'instruction (ex : `add` $\rightarrow$ 0x00).
`int parse_register(const char *reg_str)`	**Parsing de Registre.** Extrait le numéro entier du registre à partir de la chaîne de caractères (ex : `"$1"` $\rightarrow 1$). Retourne -1 en cas de format invalide.
`int find_label_pc(const char *label_name, ...)`	**Résolution de Label.** Recherche une étiquette dans la table des labels (Label*) et retourne l'adresse du **Program Counter** (`pc`) correspondante. Essentiel pour le calcul des décalages de branchement.

2 Fonctions d'Assemblage (Logique Deux-Passes) - Suite

Cette section détaille le rôle de chaque fonction dans le processus d'assemblage en deux étapes.

Fonction	Rôle et Détail
`first_pass(...)`	**Première Passe (Collecte des Labels).** Cette fonction lit le fichier d'entrée une première fois. Elle : — Détermine quelles lignes sont des instructions et quelles lignes définissent des labels. — Calcule l'adresse du **Program Counter (PC)** pour chaque instruction. — Stocke l'adresse (PC) de chaque label dans la `label_table`. Cette étape est indispensable pour que les instructions de branchement (`beq`, `bne`, etc.) puissent calculer leur décalage (*offset*) lors de la deuxième passe. — Stocke les lignes nettoyées dans `source_rows` pour une lecture séquentielle lors de la deuxième passe.
`assemble_instruction(...`	**Moteur de Traduction (Instruction Unique).** C'est la fonction centrale qui prend une instruction assembleur (une ligne) et la traduit en un mot de **32 bits binaire** (`uint32_t`). Elle : — Découpe la ligne avec `strtok` pour extraire l'Opcode et les opérandes. — Utilise `get_opcode`, `parse_register`, et `find_label_pc` pour obtenir les valeurs numériques. — Effectue le décalage de bits (`«`) pour positionner correctement les champs (Opcode, r_1, r_2, r_3/Cst/Label [11 :0]) en fonction du type d'instruction (R, I, Mémoire, Branchement). — Pour les branchements, elle calcule le **décalage relatif** (Target_PC − (Current_PC + 1)).
`second_pass(...)`	**Deuxième Passe (Écriture du Fichier Binaire).** Cette fonction est l'orchestrateur principal. Elle : — Appelle `first_pass` pour remplir la table des labels. — Parcourt les lignes du code source et, pour chaque instruction, appelle `assemble_instruction` pour obtenir le code machine de 32 bits. — Écrit le mot de 32 bits directement dans le `output_file` en utilisant `fwrite`. Cela garantit un fichier de code binaire pur (4 octets par instruction). — Gère l'incrémentation du compteur PC pour le suivi de l'adresse actuelle.

3 Fonction Principale

Cette fonction est le point d'entrée du programme et orchestre l'ensemble du processus d'assemblage.

Fonction	Rôle et Détail
`main(int argc, char *argv[])`	**Point d'Entrée du Programme.** Gère l'exécution et l'interface avec le système d'exploitation. Elle assure les étapes suivantes : — **Gestion des Arguments :** Vérifie le nombre d'arguments (`argc != 3`) pour s'assurer que le chemin du fichier d'entrée (.asm) et celui du fichier de sortie (.bin) sont fournis. — **Ouverture des Fichiers :** Ouvre le fichier d'entrée en lecture (`"r"`) et le fichier de sortie en **écriture binaire** (`"wb"`). L'ouverture en mode binaire est essentielle pour garantir que le fichier de sortie contient 4 octets par instruction sans ajout de caractères de nouvelle ligne spécifiques au système d'exploitation. — **Lancement de l'Assemblage :** Appelle la fonction `second_pass` pour démarrer le processus de traduction en deux étapes. — **Nettoyage :** Ferme tous les pointeurs de fichiers ouverts. — **Statut :** Affiche un message de réussite en cas d'assemblage correct.