Programmation assembleur

Domaine Ingénierie et Architecture
Filière Informatique et systèmes de communication
Orientation Systèmes informatiques embarqués (ISCE)
Mode de formation Temps partiel/En emploi

Informations générales

Nom : Programmation assembleur
Identifiant : ASM
Année académique : 2021-2022
Responsable : Daniel Rossier
Charge de travail : 120 heures d'études
Périodes encadrées : 64 (= 48 heures)

Semestre	E1	S1	S2	E2	S3	S4	E3	S5	S6	E4	S7	S8
Cours						32						
Laboratoire						32						

Connaissances préalables recommandées

L'étudiant-e doit connaître et savoir utiliser les notions suivantes :

- l'architecture d'un ordinateur (interfaces, bus d'adresse/données, CPU, interruptions matérielles)
- la programmation C de bas niveau (manipulation de pointeurs)

Objectifs

A l'issue de cette unité d'enseignement, l'étudiant-e sera capable de :

- Connaître les variantes d'architecture de processeur et l'impact sur le jeu d'instructions
- Concevoir une application en langage C manipulant les adresses mémoires et accès aux registres I/O
- Comprendre les notions d'environnement croisé (chaînes d'outils croisés, environnement hôte-cible)
- Comprendre les mécanismes d'un fichier de type *Makefile* et être capable de concevoir un tel fichier (simple)
- Comprendre les notions d'assembleur et caractéristiques d'un processeur
- Connaître et savoir utiliser les principales directives de compilation
- Comprendre le jeu d'instructions d'un processeur, incluant les instructions de traitement, d'accès mémoire et de branchement
- Comprendre la notion de macro-instruction et de pseudo-instruction
- Comprendre la notion de mode d'adressage
- Comprendre les mécanismes de gestion d'une pile et analyser son comportement
- Comprendre les appels de fonction, conventions d'appel (prologue/épilogue)
- Décrire le rôle d'un compilateur et du code généré (code objet)
- Connaître et savoir utiliser quelques instructions du jeu étendu x86 et ARM pour les opérations arithmétiques performantes grâce à la notion de vecteurs et d'instructions SIMD

Contenu et formes d'enseignement

Répartition des périodes indiquée à titre informatif.

Cours: 32 périodes

- Objectifs de la prog. assembleur, architecture des processeurs	2
- Programmation système en C, environnement hôte-cible	2
- Chaîne de compilation croisée (cross-toolchain), introduction aux fichiers Makefile, trraitement d'un Makefile	2
- Processeurs et registres x86, processeurs et registres ARM, introduction aux langages d'assemblage, contenu d'un fichier source	2
- Directives de compilation, jeu d'instructions x86, jeu d'instructions ARM	2
- Instructions de traitement sur ARM, instructions de traitement sur x86	2
- Macro-instructions, instructions d'accès mémoire sur ARM	2
- Pseudo-instructions sur ARM, instructions d'accès mémoire sur x86	2
- Modes d'adressage sur ARM, modes d'adressage sur x86	2
- Instructions de branchement sur ARM, instructions de branchement sur x86	2
- Gestion de la pile sur x86, gestion de la pile sur ARM	2
- Appels de fonction, fonctions imbriquées, contextes de fonction	2
- Conventions ABI, instructions SIMD, instructions x86 SSE2	2
- Instructions SIMD ARM NEON/VFP	2
- 3 travaux écrits	4

Laboratoire: 32 périodes

- Environnement Eclipse & gdb, manipulations en langage C, inspection des binaires, opérations arithmétiques et logiques en C	4
- Exercices de déploiement sur ARM & x86, utilisation du debugger, manipulations avec des fichiers Makefile	4
- Utilisation des directives de compilation, opérations arithmétiques et logiques élémentaires, conversion d'une chaîne de caractères hexadécimaux en décimal, somme de contrôle par mot de parité	4
- Assembleur « inline » en C avec GCC, manipulations avec un outil d'exploration de compilation (compiler explorer)	4
- Bruit « poivre et sel « (traitement d'image en x86), macro/pseudo-instructions et modes d'adressage ARM	4
- Distance de Hamming avec taille arbitraire de tableaux, parcours d'un arbre binaire trié	4
- Chiffrement de César, nombres premiers isolés	4
- ABI ARM AAPCS, accélération d'un filtre à réponse impulsionnelle finie	4

Bibliographie

- "ARM System Developer's Guide", 1st Edition, Andrew Sloss Dominic Symes Chris Wright, 2004
- "Assembly Language Programming in GNU/Linux for IA32 Architectures", Rajat Moona, 2007

Contrôle de connaissances

Cours : l'acquisition des matières de cet enseignement sera contrôlée au fur et à mesure par des tests et des travaux personnels tout au long de son déroulement. Il y aura au moins 3 tests d'une durée totale d'au moins 2 périodes.

Laboratoire : Ils seront évalués sur la base des rendus de code par étape de 2 ou 3 laboratoires, à 3 reprises au minimum.

Calcul de la note finale

Note finale = moyenne cours x 0.67 + moyenne laboratoire x 0.33