UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES
université PARIS-SACLAY

IUT Informatique 1A - FI S1
Mécanismes de Bas Niveau
2024 – 2025

Alain Oster / Christophe Chassat
alain.oster@uvsq.fr/christophe.chassat@cnrs.fr

TP 3 - Stockage et codage

Préambule

Démarrer votre station de travail sous Windows (ou sous l'OS de votre choix). Ouvrir votre navigateur à l'adresse locale (sans espace) :

$http://192.168.25.25:8000$

Exercice 1 Big Endian / Little Endian pour stocker des données en mémoire

Rappel : lorsqu'on utilise l'instruction n ASSIGN32 0x4 (qui déclare et affecte la variable n=4), la valeur 0x4 est stockée en mémoire.

Stocker 0x12345678 et observer ce qu'on obtient en mémoire : | | | | |

Stocker 0xAABBCCDD et observer qu'on obtient en mémoire : | | | | |

Peut on dire que le microprocesseur utilise le stockage **Little Endian** pour les données ?

Exercice 2 Big VS little Endian pour stocker les programmes en mémoire

le code binaire original de MOV R1, R0 est E1 A0 10 00

Ecrire un programme qui contient l'instruction MOV R1, R0 (écrire l'instruction plusieurs fois et/ou utiliser le mode d'exécution ligne par ligne).

Quel est le codage en mémoire :

00	10	A0	E1	**Little Endian** ?

E1	A0	10	00	**Big Endian** ?

Compléter le tableau :

instruction	code en mémoire	code original
MOV R1, R0		E1 A0 10 00

Peut on dire que le microprocesseur utilise le stockage **Little Endian** pour les programmes ?

Exercice 3 : Tableau d'instructions

En s'inspirant de la méthode utilisée à l'exercice précédent compléter le tableau suivant :

instruction	code en mémoire	code original
MOV R1, R0	00 10 A0 E1	E1 A0 10 00
MOV R2, R0		
MOV R2, R1		
MOV R3, R1		
ADD R0, R0, R0		
ADD R0, R0, R1		
SUB R0, R0, R1		
CMP R0, R1		
CMP R0, R2		
CMP R0, R3		

Exercice 4 Décodage

Une partie du programme en assembleur a été effacée : compléter les parties manquantes grâce au code stocké en mémoire et à l'aide du tableau précédent. Ecrire le programme complet.

```
1  SECTION INTVEC
2
3  B main
4  SECTION CODE
5
6  main
7
8  LDR R0, note ; R0 <- note
9  LDR R1, bonus ; R1 <- bonus
10 ??????????????
11 MOV R3, #0x14 ; 0x14 correspond à 20 en décimal
12 ??????????????
13 BGT seuil ; saute au label seuil si ??????
14 B continue
15 seuil
16 MOV R0, #0x14 ; 0x14 correspond à 20 en décimal
17 continue
18 STR R0,note
19 fin
20 B fin
21 SECTION DATA
22 note ASSIGN32 0x12 ; declaration et affectation de la variable
23 bonus ASSIGN32 0x4 ; declaration et affectation de la variable
```

Mémoire

addr	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0x00000000	1E	00	00	EA	--	--	--	--	--	--	--	--	--	--	--	--
0x00000010	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000020	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000030	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000040	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000050	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000060	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000070	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000080	78	0F	9F	E5	78	1F	9F	E5	01	00	80	E0	14	30	A0	E3
0x00000090	03	00	50	E1	00	00	00	CA	00	00	00	EA	14	00	A0	E3
0x000000a0	58	0F	8F	E5	FE	FF	FF	EA	--	--	--	--	--	--	--	--

Que fait ce programme ? On pourra le tester avec note=15 bonus=2 puis avec note=18 bonus=4.