Architecture des Ordinateurs - Projet 2020

- Le projet est à réaliser par binôme
- Les consignes de remise sont spécifiées sur *moodle*.
- Toute copie d'une partie du projet entrainera la note 0 pour l'ensemble des étudiants présentant un projet similaire
- Tout projet non rendu entrainera la note 0.
- Vous rendrez le fichier *deasm_fonction.c*, le fichier *deasm.asm* exécutable dans Mars, les fichiers du processeurs *mips.circ*, *cmd.dig*, *ual.dig decodeur64.dig encodeur16.dig* et les fichiers hexadécimaux *Inst.hex* du programme et *data.hex* des données du processeur.

Le but du projet est de réaliser un dé-assembleur MIPS simplifié.
Le programme à dé-assembler sera contenu dans un tableau.

Listing 1: tableaux

```
%char reg[32][4]={ "$0",...}; // nom des registres
char nomImm[64][8]={"X","X","j","jal",...}; // instructions 3 registres
char nom3reg[64][8]={  "sll",...}; // Instructions 3 registres
unsigned pgm[]={ unsigned pgm[]={0X2044000a,0x0c100007,
0x00402021,0x20020001,0x0000000c,0x2402000a,0x0000000c,0xafa40000,
0x23bdfffc,0x03bd1026,0x00441020,0x2084ffff,0x1480fffd,0x23bd0004,
0x8fa40000,03e00008};}; // Programme de test
```

Nous devons pour ce programme obtenir l'affichage suivant :

```
addi $a0,$v0,10
jal 0x40001C
sub $a0,$v0,$0
addi $v0,$0,1
syscall
addiu $v0,$0,10
syscall
sw $a0,0($sp)
addi $sp,$sp,-4
xor $v0,$sp,$sp
addu $v0,$v0,$a0
addi $a0,$a0,-1
bne $0,$a0,-2
addi $sp,$sp,4
lw $a0,0000($sp)
jr $ra
```

1 programme C - 5 points

Créer les fonctions suivantes

Listing 2: deasm.h

```
#include <stdio.h>

void strAdd(char *src,char *dest); // Ajouter src a la fin de  destination
void strCpy(char *scr,char*des); // Copie source dans destination

unsigned getRegName(unsigned num,char* name); // Contiendra char regName[32][4] ={"$0","$at"....} ;
    retourne 0 si OK, 1 sinon
unsigned getRs(unsigned codeInst,unsigned *numRs, char* name); // Numero registre et chaine
    contenant nom symbolique
unsigned getRt(unsigned codeInst,unsigned *numRt, char* name); // retourne 1 si erreur
unsigned getRd(unsigned codeInst,unsigned *numRd, char* name);

void utoha(unsigned nbr,char *chaine); // chaine represantant valeur hexa de nbr  avec un minumum de
     caracters
```

```
void itoa(int nbr,char *chaine); //chaine represantant valeur decimale de nbr


void getImmS16(unsigned codeInst,int *imms,char *chaine); // valeur imediate signee 32  bits (en
    decimal)
void getImmNs16(unsigned codeInst,unsigned *immNs,char *chaine); // valeur immediate non signee
    16->32bits en hexa debutant par 0x
void getImmNs26(unsigned codeInst,unsigned *immNs,char *chaine); // valeur immediate non signee
    26->32bits  avec decalage a gauche de 2 debutant par 0x

unsigned getInstructionName(unsigned codeInst,unsigned *Co,int *Nf,char *name);
//contiendra  char nomImm[64][8]={"?","?","j","jal".... a completer }  Vous choisirez un code
    particulier pour les instructions non traitees
//et char nom3reg[64][8]={"sll",.... a completer}
//Retourne 1 si instruction non trouvee 0 sinon

unsigned decodeInstruction(unsigned codeInst,char *inst);
// Un seul espace entre nom et operandes pas d'autres espace
// Format 3 reg -> nom $rd,$rs,$rt
// Format 3 reg shift -> nom $rd,$rt,$rs
// Format 3reg shamt -> nom $rt,$rs,shamt (en decimal)
 // Format Immediate arithmetique  ual, nom $rt,nom $rs,ImmS (en decimal)
// Format Immediate logique ual, nom $rt,nom $rs,ImmNS (en hexa 0x....)
// format chargement, rangement nom $rt,ImmS($rt)
// format branchement nom $rs[,$rt],ImmS (en decimal - nombre d'instruction par rapport au
    branchement)
// format saut jr $rs
// format jal -> jal 0x..... (valeur de l'adresse de la fonction en hexa )
// Instruction inconnue -> "#Unknown"
// Retourne 0 si instruction connue1 sinon

unsigned decodePgm(unsigned *pgm,unsigned taille,char [100][100]);
// decode les taille instructions du programme  (max 100 ) dans un tableau de 100 caracteres par
    instruction
```

La fonction main() appelle decodePgm et affichera les chaînes de caractères contenues dans le tableau résultat. La fonction *decodeInstruction* prendra en compte toutes les instructions du cours . Le fichier *deasm.h* à inclure au début des fonctions est disponible sur *moodle*. Un jeu de test vous sera donné dans quelque temps. Vous m'utiliserez aucune fonction des bibliothèques C.

2 Programme Assembleur - 5 points

Transformer le programme ci-dessus en instructions MIPS à l'aide du compilateur en ligne et tester le dans MARS. Télécharger les code hexadécimaux depuis MARS dans deux fichiers nommés data.hex et instruction.hex au format *Intel Hex format*.

3 Digital - 5 points

Charger le programme ci-dessus (instructions et données) dans digital et tester le. Vous rendrez le processeur avec les fichiers hexadécimaux.

4 Amélioration

A faire seulement si toutes les questions précédentes fonctionnent correctement.

— Gérer plus d'instructions que celle vues en cours.