|  |  |  |
| --- | --- | --- |
| ttps://www.univ-rennes1.fr/sites/www.univ-rennes1.fr/themes/ur1_core/logo.png | **ARC** CC1 ILM | **Février 2022** |

* Durée de l'épreuve : 1h00 heures
* Polycopiés de cours autorisés
* Calculatrices non connectées autorisées.
* Le barème est donné à titre purement indicatif.

**Les réponses doivent être portées sur ce document, sous peine d’être ignorées par le correcteur.**

### Exercice 1 : Jeu d'instructions Nios II (5 points)

On s’intéresse à un processeur NIOS-II qui exécute les instructions d’un programme représenté ci-dessous. L’état de la mémoire au début de l’exécution du programme est représenté dans le tableau ci-après.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **movia r2,0x12ABCD**  **addi r3,zero,0x8FFF**  **stb r2,0x6(zero)**  **ldh r4,0x2(zero)**  **srai r5,r3,8** | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | **00** | **01** | **10** | **11** | | **0x000** | **0xF1** | **0x23** | **0xC3** | **0x84** | | **0x004** | **0x40** | **0x53** | **0x9D** | **0x62** | |

**Q1**: Complétez les tables ci-dessous en précisant les valeurs des registres ainsi que l’état de la mémoire à l’issue de l’exécution de ce programme.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **r2** | **0000FFFF** | | **r3** | **FFFF8FFF** | | **r4** | **FFFF84C3** | | **r5** | **FFFFFF8F** | | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | **00** | **01** | **10** | **11** | | **0x000** | **0xF1** | **0x23** | **0xC3** | **0x84** | | **0x004** | **0x40** | **0x53** | **0xFF** | **0x62** | |

**Q2** : Pouvait-on utiliser l'instruction **movi** à la place de **movia** dans le programme ci-dessus ? Si non, expliquez pourquoi.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q3**: Que se passe-t-il lors de l'exécution de l'instruction **ldw r4,0x1(zero)** ? Expliquez pourquoi.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Exercice 2 : programmation assembleur (10 points)

On s'intéresse ici à la traduction du code C ci-dessous vers le langage machine NIOS II

|  |
| --- |
| **struct** node {  **int**\* data;  **struct** node\* next;  };  **int** tab[] = {1,45,234};  **struct** node root = { .data=tab, .next=NULL};  **struct** node\* current = &root; |

**Q1** : Donnez la taille (en octets) de l'espace mémoire occupé par chacune des variables globales **tab**, **root** et **current** (vous justifierez votre réponse).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q2** : Complétez les directives d'assemblage NIOS-II permettant d'allouer et d'initialiser les variables globales **tab**, **root** et **current**. Déduisez en les valeurs associées aux identificateurs **\_tab**, **\_root**, **\_current** ainsi que le contenu de la mémoire au début du programme.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| .data  \_tab: .word 1  .word 45  .word 234  \_root: .word tab  .word 0  \_current: .word root | |  |  | | --- | --- | | Ident | valeur | | **\_tab** |  | | **\_root** |  | | **\_current** |  | | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | **0x0** | **0x1** | **0x2** | **0x3** | | **0x000** |  |  |  |  | | **0x004** |  |  |  |  | | **0x008** |  |  |  |  | | **0x00C** |  |  |  |  | | **0x010** |  |  |  |  | | **0x014** |  |  |  |  | |

**Q3** : Traduisez (en langage d'assemblage NIOS-II) l'instruction ci-dessous.

|  |  |
| --- | --- |
| current=NULL; | stw zero, \_current(zero) |

**Q4** : Traduisez (en langage d'assemblage NIOS-II) l'instruction ci-dessous (on fera l'hypothèse que la valeur associée à **\_current** est codée sur 16 bits)

|  |  |
| --- | --- |
| root.next = current; | ldw r2, \_current(zero)  stw r2, \_next+4(zero) |

**Q5** : Traduisez (en langage d'assemblage NIOS-II) la séquence d'instruction ci-dessous.

|  |  |
| --- | --- |
| current -> data[4] = -1; | ldw r2, \_current(zero) ; r2 =current  ldw r3, 0(r2) ; r3 = current->data  movi r4,-1  stw r4, 16(r3) ; \*(current->data+4)=-1; |

**Q6** : Traduisez (en langage d'assemblage NIOS-II) la fonction ci-dessous, en vous assurant que la fonction (i) respecte les conventions d'appels vues en cours et (ii) n'ait pas d'effet de bord sur les registres de la machine.

**NB**. : vous pourrez utiliser au choix les conventions simplifiées utilisées en CM ou bien celles de **gcc** utilisées en TD/TP.

|  |  |
| --- | --- |
| **void** q6(**struct** node \*p,**int** i) {  if (p!=NULL)  p->data[i]=0;  } | q6:  subi sp, sp, 8  stw r4,(sp)  stw r5,4(sp)  beq r4,zero, end  ldw r4,(r4)  srli r5,r5,2  add r5, r5, r4  movi r4,0  stw r4,(r5)  end:  ldw r5,4(sp)  ldw r4,(sp)  addi sp, sp, 8  ret |

### Exercice 3 : rétro-ingénierie de code machine (6 points)

Soit la fonctions C **mystery** dont le prototype est donné ci-après.

**int\*** mystery(**int\*** in**, int** size)

On dispose du code machine de la fonction **mystery**, et on souhaite reconstruire son code C, sachant qu'il a été produit par le compilateur gcc (et suit donc les conventions établies en TD/TP). Donner le code C correspondant.

**Attention, le code C devra être syntaxiquement correct et aussi proche que possible de ce qu'aurait écrit un programmeur humain.**

|  |  |
| --- | --- |
| Assembleur | Code C |
| **mystery:**  **addi sp, sp, -12**  **stw r16, 0(sp)**  **mov r16, r4**  **slli r4, r5, 2**  **stw r17, 4(sp)**  **stw ra, 8(sp)**  **mov r17, r5**  **call malloc**  **mov r4, r16**  **movi r5, 1**  **addi r17, r17, -1**  **.L2:**  **addi r4, r4, 4**  **blt r5, r17, .L3**  **ldw ra, 8(sp)**  **ldw r17, 4(sp)**  **ldw r16, 0(sp)**  **addi sp, sp, 12**  **ret**  **.L3:**  **ldw r7, 4(r4)**  **ldw r3, -4(r4)**  **slli r6, r5, 2**  **addi r5, r5, 1**  **add r3, r3, r7**  **ldw r7, 0(r4)**  **add r6, r2, r6**  **slli r7, r7, 1**  **add r3, r3, r7**  **srli r3, r3, 2**  **stw r3, 0(r6)**  **br .L2** | |  | | --- | | **int\*** mystery(**int[]** in**, int** size) { | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | | }  **void\* malloc(int);**  **int\* mystery(int\* in, int size) {**  **int i,j;**  **int\* out = (void \*)malloc(sizeof(int)\*size);**  **for (i=1; i<size-1;i++) {**  **out[i] = (in[i-1] + 2\*in[i]+ in[i+1])/4;**  **}**  **return out;**  **}** | |