République algérienne démocratique et populaire

Ministère de l’enseignement supérieur et de la recherche scientifique

Université de Hassiba Ben Bouali - Chlef
Faculté des sciences exactes et de l'informatique
Département de l'Informatique

Projet TP d’Architecture : **MIPS**

- Présenté par Bennabi Mohammed Malek (Groupe 02)

Sous la supervision du professeur Bouheraoua Fayçal

Brève Définition :

L'architecture MIPS(**M**icroprocessor.without **I**nterlocked **P**ipeline **S**tages) est une architecture de processeur de type RISC (Reduced Instruction Set Computer) développée par la société MIPS Computer Systems, basée en Californie.

Les processeurs fabriqués selon cette architecture ont surtout été utilisés dans les systèmes SGI. On les retrouve aussi dans plusieurs systèmes embarqués, comme les ordinateurs de poche, les routeurs Cisco et les consoles de jeux vidéo (Nintendo 64 et Sony PlayStation, PlayStation 2 et PSP).

Vers la fin des années 1990, on estimait que les processeurs dérivés de l'architecture MIPS occupaient le tiers des processeurs RISC produits.

De nos jours cette architecture reste populaire dans le marché de l'informatique embarquée où elle subit une intense concurrence de la part de l'architecture ARM.

Enoncé du TP :

Écrire un programme en assembleur mips déclarant un tableau initialisé par 10 entiers et recherchant le plus petit élément.

Explication de la Solution Envisagée :

On va dans un premier temps déclarer un tableau d'entiers et l'initialiser avec 10 nombres, après on va faire un tri de ce tableau en utilisant la méthode du tri par Bulle, pour pouvoir par la suite obtenir un tableau trié par ordre croissant, et logiquement dans un tableau trié par ordre croissant le plus petit élément est le premier élément du tableau donc on va l'afficher.

Le Principe du Tri par Bulle :

Le principe du tri à bulles est de comparer deux à deux les éléments t[i] et t [i+1] consécutifs d'un tableau et d'effecteur une permutation si t[i] > t [i+1]. On continue de trier jusqu'à ce qu'il n'y ait plus de permutation

Pour ce tableau de 10 éléments au début j=10. A la fin de chaque étape la partie du tableau de t [j+1 : 10] est constituée d'éléments bien placés et tous supérieurs aux éléments de t [0 : j-1].

- Pour les éléments i, de 0 à j-1 comparer et permuter successivement (si nécessaire) les éléments t[i] et t [i+1].
- Faire j = j -1, recommencer ensuite
- Arrêt de l'algorithme lorsque j = 0.

```
.data
t:  .word 10, 9, 38, -7, -128, 5, -4, 0, 2, -8

.text
sort :
   li $t0, 0
   li $t6, 10
loop :
   la $t1, t
   beq $t0, $t6, exitLoop
   sub $t5, $t6, $t0
   addi $t5, $t5, -1
   li $t4, 0 #j compteur
   addi $t0, $t0, 1
   j pass_loop

pass_loop:
   beq $t4, $t5, loop
   lw $t2, 0($t1)
   lw $t3, 4($t1)
   bgt $t2, $t3, swap
   j next
swap :
      sw $t2, 4($t1) #   t2 := [i+1]
      sw $t3, 0($t1) # t3 := [i]
      j next
next:
   lw $t2, 0($t1)
   addi $t1, $t1, 4
   addi $t4, $t4, 1
   j pass_loop

exitLoop:
   li $v0, 1
   move $a0, $t2
   syscall
```

Explication détaillée de l'exécution:

.data Spécifie le champ variable

t: .word 1, 6, -8, 18, 0, 17, 3, 9, 48, -85 la déclaration d'un tableau et initialisation de 10 entiers et word permet de stocker les valeurs sous forme de mots de 32 bits

.text Spécifie le segment text qui contient les instructions

Sort : Étiquette

li $t1, 0 charger Immédiatement : initialiser $t1 avec la valeur 0 qui sera considérée comme la variable i du compteur i

li $t6, 10 charge $t6 avec la valeur 10 qui est la taille du tableau

---

loop :

la $t1, t charger l'adresse : mettre l'adresse du tableau t dans le registre t1

beq $t0, $t6, exitLoop Branchement immédiat a exit Loop si le contenue de t6 est égal au contenue de t0

sub $t5, $t6, $t0 soustraction : le registre t5 reçoit le résultat de la soustraction du contenue de t6 et le contenue de t0

addi $t5, $t5, -1 Addition Immédiate : le registre t5 reçois le résultat de l'addition du contenue de t5 et -1

Parce qu'à chaque passage la longueur du sous-tableau deviendrait plus petite au fur et à mesure que les éléments après le sous tableau deviendront triés donc on décrémente de 1

li $t4, 0 charger le registre t4 immédiatement avec la valeur 0 qui sera le compteur j de la boucle

addi $t0, $t0, 1 addition immédiate de t0 et 1 donc incrémentation de t0

j pass_loop Saut inconditionnel : sauter vers l'adresse de l'étiquette pass Loop

---

pass_loop: Etiquette

beq $t4, $t5, loop Branchement immédiat à loop si le contenue du registre t4 et t5 sont égaux

lw $t2, 0($t1) Charger le mot: charger le registre t2 avec le contenue de l'adresse du mot mémoire du premier élément i indexé

lw $t3, 4($t1) charger le registre t3 avec le contenue de l'adresse du mot mémoire de l'élément suivant i+1 indexé

bgt $t2, $t3, swap Brancher si supérieur strictement : Brancher à swap si le contenue de t2 est supérieur du contenue de t3

j next sauter vers l'adresse de l'étiquette next

---

swap : Étiquette

sw $t2, 4($t1) Stocker le mot : stocker le contenue de t2 dans l'adresse mémoire du mot de l'élément i+1 indexé

sw $t3, 0($t1) stocker le contenue de t3 dans l'adresse mémoire du mot de l'élément i indexé

j next sauter vers l'adresse de l'étiquette next

---

next: Étiquette

lw $t2, 0($t1) charger le registre t2 avec le contenue de l'adresse du mot mémoire de l'élément i indexé

addi $t1, $t1, 4 addition immédiate de t1 et 4 donc incrémentation de t1 par 4

addi $t4, $t4, 1 addition immédiate de t4 et 1 donc incrémentation de t4

j pass_loop sauter vers l'adresse de l'étiquette pass loop

---

exitLoop: Etiquette

li $v0, 1 charger le registre v0 immédiatement avec la valeur 1 pour afficher un entier on utilise le service 1

move $a0, $t2 déplacer : charger le registre argument a0 avec le contenue du registre t2 qui contient la valeur désirée (minimum du tableau)

syscall pour afficher la valeur de t2

Déroulement du programme :

On initialise un tableau de 10 nombres aléatoires, le programme commence par la boucle Tant que ($t0 != $t6) Et à chaque fois soustrait la taille du sous tableau et incrémente le compteur i $t0 et passe a pass_loop pour comparer le compteur j $t4 et la taille du sous tableau $t5, s'ils sont égaux il rebranche a loop sinon il charge t[i] $t2 et t[i+1] $t3 et il les compare , si t[i] est inférieur ou égal à t[i+1] il branche à next sinon il va brancher à l'étiquette swap qu'elle va aussitôt inter changer les valeurs t[i] :=t[i+1] et t[i+1] :=t[i] et elle va brancher directement à next qui va passer à l'adresse de l'élément suivant i :=i+1 et i+1 :=i+2 et branche à pass_loop pour refaire les étapes précédentes jusqu'à ce que i==taille du tableau $t0==$t6 à ce moment il branche à exit_loop ce qui signifie que le tableau est totalement trié donc exit loop va directement charger l'adresse du premier élément du tableau dans le registre d'argument et elle va l'afficher à l'utilisateur (Dans mon cas le minimum est -128).