ES201 - Architecture de Microprocesseurs

Travail Dirigée

12 septembre 2024

Guilherme Nunes Trofino
2022-2024

Table des matières

1. Introduction

Repository Hello! My name is Guilherme Nunes Trofino and this is my LaTeX notebook of ES201 - Architecture de Microprocesseurs that can be found in my GitHub repository : https://github.com/tr0fin0/classes_ensta.

Disclaimer This notebook is made so it may help others in this subject and is not intend to be used to cheat on tests so use it by your on risk.

Suggestions If you may find something on this document that does not seam correct please reach me by e-mail : guitrofino@gmail.com.

1.1. Information Matier

Référence microprocesseur faire une bouque infinite modele de machine von neummann mode d'exécution générale si il n'y a pas des taches à faire qu'est-ce que le processeur faire pendant ce temps : If the hardware doesn't make allowance for this, then the CPU will have to run useless instructions until it is needed for real work
me mets en reve et me reveie faire surplace, bloque sur le meme adresse memoire, rester en bloque sur lui meme adresse PC est program counter necessaire pour savoir ce que le programme doit executer
instruction memory operation instruction decode par l'unité de contrôle control units
methodologie de conception unité de calcule, machine de miller machine de ... états finits multiplexeur registres unites fonctionnelles mémoire
ALU R instructions
000 and 001 or 010 add 110 sub 111 slt
temps d'accès et temps de propagation plus grand mémoire plus grand le temps d'accès et propagation c'est une contrainte, plus de temps moins grand CPU
dans une processeur monocycle cache données et cache instruction le chemin le plus long est le chemin critique et denote le temps de cycle
inconvenienst du monocycle toutes les instructions worst case execution time, si minimizer le systeme sera plus efficace
avantage connait le temps pour executer les instructions
modele de séquencement
le decoupage ne peut pa passer le temps de memoirasion aléas de structures structural hazards aléas de contrôle control hazards aléas de données data hazards

Question 1

Résolution. a

```
    switch:
    andi  s6, s6, 0         // counter = 0

    case0:
    bne  s6, s5, case1     // k = 0
    add  s0, s3, s4        // f = i + j

    case1:
    addi  s6, s6, 1
    bne   s6, s5, case2     // k = 1
    add   s0, s1, s2        // f = g + h

    case2:
    addi  s6, s6, 1
    bne   s6, s5, case3     // k = 2
    sub   s0, s1, s2        // f = g - h

    case3:
    addi  s6, s6, 1
    bne   s6, s5, end       // k = 3
    sub   s0, s3, s4        // f = i - j

    end:
    j ret
```

Le code précédent sera scalable, donc il peut être facilement changer pour ajouter des nouvelles conditions dans le case.

Ici il y a le problème de temps d'exécution pas constant, ça veut dire qu'il y aura des temps d'exécution différent par chaque cas car il y aura besoin de plusieurs essayes avant d'arriver à le bon résultat.

Question 2

Résolution. a

```
    andi s1, s1, 0        // sum = 0
    andi s2, s2, 0        // i = 0
    addi s3, zero, 500    // j = 500
    andi s4, s4,0         // k = 0
    addi t2, zero, 1000   // 1st loop upper bound = 1000
    andi t3, s6, 0        // 2nd loop lower bound = 0
    addi t4, zero, 300    // 3rd loop upper bound = 300

    loop1:
    loop2:
    loop3:
    add  s1, s1, s2       // sum = sum + (i);
    add  s1, s1, s3       // sum = sum + (i + j);
    add  s1, s1, s4       // sum = sum + (i + j + k);
    addi s4, s4, 10       // k = k + 10

    bne  s4, t4, loop3    // if (k < 300) goto loop3, else execute 2nd loop
    andi s4, s4,0         // reinitialize k = 0
    subi s3, s3, 1        // j--

    bne  s3, t3, loop2    // if (j > 0) goto loop2, else execute 1st loop
    addi s3, zero, 500    // reinitialize j = 500
    addi s2, s2, 1        // i++

    bne  s2, t2, loop1    // if (i < 1000) goto loop1, else end program
    exit:
```

Chaque register devrait avoir un $ dans la convection de cette matière mais ici ce caractère a été élevé car le style du code ne marchait pas.

Ici, en fait, au lieu d'avoir 3 couples c'est plus facile pour comprendre qu'il y a un grand couple avec 3 conditions necessaires à attendre pour que le code s'arrêt.

Question 3

Résolution. a

```
leaf:

subi  sp, sp, 12   // adjust stack to make room for 3 items
sw    s2, 8(sp)    // save register s2 for further use
sw    s1, 4(sp)    // save register s1 for further use
sw    s0, 0(sp)    // save register s0 for further use

add   s2, a0, a1   // register s2 contains g + h
add   s1, a2, a3   // register s1 contains i + j
sub   s0, s2, s1   // f = s2 - s1 which is (g+h) - (i+j)
add   v0, s0, zero // returns f (v0 = s0 + 0)

lw    s0, 0(sp)    // restore register s0 for caller
lw    s1, 4(sp)    // restore register s1 for caller
lw    s2, 8(sp)    // restore register s2 for caller
addi  sp, sp, 12   // readjust the stack pointer

jr ra              // jump back to calling routine (ra = return address)
```

Les registres a0, ..., a7 (il faut confirmer la quantité précise) sont les endroits où les entrées et sortis d'une fonction seront estoquées.

Comme on va utiliser les registres s0, s1 et s2 pour faire les calculs de cette fonction il faut qu'on les estoquées avant les données sur ses registres car une autre fonction pourrait les utiliser et on ne peut pas les déranges.

Question 4

Résolution. a

```
    ssh -X login@salle.ensta.fr
    ssh -Y

    cp -r /home/g/gac/ES201/TPs/TP2
    cp -r /home/g/gac/ES201/tools/graph/SSCA2v2-C
```

il faut chercher comment faire l'utilisation du code