UNIVERSITAT DE BARCELONA

Pràctica 2: Exercicis amb el simulador Ripes

Alejandro Guzman

18 març 2022

ÍNDEX

1 OBJECTIUS

L'objectiu principal de la segona pràctica de la assignatura és familiaritzar-se amb el conjunt de registres que hi ha a la CPU (tant registres específics com de propòsit general) utilitzant el simulador d'exercicis per a la realització d'un conjunt d'exercicis tant a classe com treball autònom.
En les sessions teoricopràctiques repassarem els conceptes d'instruccions i els seus formats i modes d'adreçament, i realitzarem un conjunt d'exercicis reflexats en un apartat d'aquest informe.

2 QÜESTIONS

1 Exercici 1

Un cop acabat el programa quant val el contingut de A5?
En acabar l'execució del programa, el contingut d'A5 val el resultat de l'operació aritmètica Dada[0] + Dada[1], és a dir, A5 = 9 + 3 = 12 = **0b00000000000000000000000000001100** on aquests nombres són les dades guardades prèviament a la execució de les instruccions del programa.
La operació de la suma es repeteix un total de cinc iteracions dins del bucle, però a cada iteració el resultat que es guarda és el mateix ja que no varia el contingut dels registres A1 i A2 (què són Dada[0] i Dada[1] respectivament).

Un cop acabat el programa quant val el contingut de A6?
En acabar l'execució del programa , el contingut d'A6 val el resultat de l'operació aritmètica Dada[1] + Dada[3], és a dir, A5 = 3 + 1 = 4 = **0b00000000000000000000000000000100**. De manera anàloga a l'explicat a l'anterior apartat, la instrucció s'executa cinc cops dins del bucle però al no variar els registres A2 i A4 (què són Dada[1] i Dada[3] respectivament), llavors no varia el contingut d'A6.

Un cop acabat el programa quant val el contingut de A3?
El contingut d'A3 serveix com a variable de bucle, és a dir, abans del loop se li assigna el valor de la Dada[2] = 4 i a cada iteració disminueix el seu valor en 1, en la mateixa iteració la condició per sortir del bucle és que el contingut del registre A3 sigui zero, per tant fins no passades cinc iteracions no sortirà del bucle. A la cinquena es comprova que efectivament el contingut d'A3 és zero, però en la mateixa iteració passa per la instrucció de restar una unitat al contingut d'A3, per tant el seu valor és -1 expressat en complement a 2, és a dir, A3 = -1 = **0b11111111111111111111111111111111**.

Quines instruccions reals s'utilitzen per a les pseudoinstruccions nop i bgez? (mireu el codi màquina que es genera i consulteu la taula d'instruccions, utilitzant els camps opcode i func3/7 si cal)
Mirant el codi màquina que es genera, assegurem que les instruccions reals que s'executen per a aquestes **pseudoinstruccions** són *addi zero, zero, 0* (suma el contingut del registre x0 a l'immediat 0 i carrega-ho al registre x0, que és només de lectura i el seu contingut valdrà 0 tot i que hi vulguis carregar una altra cosa) i *bge a3, zero, loop* (salta a la posició *loop* si els continguts dels registres a3 i x0 són iguals. És a dir, si el contingut del registre a3 és 0).

En quina posición de memòria es troben els valors de Resultat? En quina posició de memòria es guarda el contingut del registre A6?

Els valors de Resultat es troben a les posicions de memòria 0x10000010 i 0x10000014 (en hexadecimal) (les adreces avancen de 4 en 4 al tractar-se d'un processador de 32 bits). La posició de memòria on es guarda el contingut del registre A6 és el contingut d'A0, que recordem que és un punter que està apuntant a la primera posició de memòria, més 20 unitats, és a dir a la posició Resultat[1] que, com hem mencionat abans, és la posició 0x10000014 expressat en hexadecimal.

Quan executem bgez, quin registre es veu afectat?
El registre bgez compara el registre origen amb el registre zero (que només s'utilitza de lectura), per tal de comprovar si l'origen té de contingut zero, per tant es veuen afectats el registre A3 i el registre zero per tal de realitzar la comparació amb el Branch, tal i com es pot veure en el mode extended del Single Cycle Risc-V Processor.

2 Exercici 2

Quina operació matemàtica està realitzant aquest codi?
Aquest codi està realitzant d'una manera implícita una multiplicació, ja que esta sumant al mateix registre el mateix nombre a cada iteració (A[1] = 3083_{10} = 0000110000001011_2), com el bucle itera 18 vegades, la operació que realitza és $3083 \cdot 18 = 55494$.

Quant val el contingut d'A0, A1 i A2 quan acaba el programa?
Quan acaba el programa el contingut d'A0 val el resultat de l'operació matemàtica explicitada en l'apartat anterior, és a dir, val A[0] = 55494_{10} = 1101100011000110_2.
El contingut del registre A1 val A[1] = 3083_{10} = 0000110000001011_2, és a dir, no canvia en tot el transcurs del programa desde que se li assigna aquest valor mitjançant una instrucció *load word*.
Finalment el contingut del registre A2 és A[2] = -1_{10} = $...1111_2$. Això és degut a que, tot i que la condició d'aturada és que el contingut d'aquest registre ha de valdre 0, després de passar aquesta condició, es repeteix una última vegada la condició de restar una unitat al contingut d'A2, per tant val -1 i no 0.

Quantes iteracions del bucle executa el codi?
El resultat final del registre A0 és 55494, com a cada iteració es suma 3083 al contingut d'aquest registre i es guarda en el mateix registre, llavors $\frac{55494}{3083} = 18$ i per tant s'executa 18 vegades.

3 Exercici 3

En aquest exercici se'ns demana implementar un algorisme que realitza unes comprovacions i executa unes instruccions subjectes a certes condicions, a continuació es mostra el codi:

```
# Definició de les entrades del programa assignant valors arbitraris en dues posicions de memòria.
.data
A: .word 56
B: .word 28

.text
main:
# Carreguem a dos registres els valors dels valors d'A i B.
la a0, A
lw a1, 0(a0)
lw a2, 4(a0)
# Comprovem si A <= B, en cas negatiu sumem els valors, en cas afirmatiu saltem a la posició de nosuma.
ble a1, a2, nosuma

suma:
add a3, a1, a2
j end

nosuma:
# En cas de que A = B sumem els valors, en cas de que no restem A - B.
beq a1, a2, multiplica
sub a3, a2, a1
j end

multiplica:
mul a3, a1, a2

# Totes operacions guarden el resultat en el registre a3.

end:
nop

```

Figura 2.1: Codi exercici 3

Expliqueu el funcionament del programa que heu implementat.
Primerament, carreguem l'entrada A al registre a1 i la B al registre a2. Si el contingut de a1 és més gran que el de a2 (A > B), els sumem i es guarda el resultat al registre a3. De no ser així, salta a la posició *nosuma* i es realitza una comprovació en forma de condicional si els continguts dels registres són iguals. En cas afirmatiu salta a la posició de memòria *multiplica* i multiplica els dos valors A i B, després els guarda en a3. En cas de ser diferents, aleshores es té A < B i realitza la resta entre ambdós valors, carregant el resultat al registre a3 (com són casos excloents es pot usar el mateix registre per guardar el resultat de les tres operacions). Un cop es realitza qualsevol de les operacions, finalitza el programa (és important la instrucció "j end" després de la suma i la diferència ja que, de no afegir-la, després de fer la suma es faría la resta i després la multiplicació).

Feu que el resultat es guardi a la posició de memòria 24h bytes després de l'inici de la secció .data.
Per realitzar el que demana l'enunciat només s'ha d'escriure una instrucció al final del codi de forma que es realitzi un *store* del contingut del registre a3 a la posició de memòria 24h bytes (un salt de 24 en hexadecimal) després de l'inici de la secció .data, és a dir, de l'adreça que guarda el registre a0.
A continuació es mostra la instrucció:

```
30 end:
31 sw a3, 0x24(a0)
32 nop
33
```

Figura 2.2: Instrucció de *store*

3 Conclusions

Aquesta pràctica ha servit per adentrar-nos més en el simulador *Ripes*, en el llenguatge assemblador i en el proceessador *Risc-V*. El primer exercici ha sigut una introducció al concepte de vectors i a l'accessibilitat a memòria mitjançant el l'adreçament de memòria de tipus relatiu, el segon exercici continua amb els mateixos conceptes de programació en llenguatge assemblador però, a part, demana a l'estudiant cambiar certa part del codi perquè ell mateix comenci a testejar i a picar codi amb aquest llenguatge. Finalment l'últim exercici és de realitzar un algorisme de forma quasi independent sense cap plantilla, la qual cosa incentiva a comprendre millor tant el simulador com el tipus de llenguatge en el qual s'està programant.
En definitiva, ha sigut una pràctica molt interessant que ens ha oferit coneixements amb cert grau de profunditat extra en relació amb els conceptes de l'assignatura i així posar en pràctica la teoria donada en les sessions teòriques.