PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2343 - Arquitectura de Computadores
Marzo 2022

Entrega el 9 de mayo hasta las 23:59

Tarea 2

Objetivos

Para esta tarea, esperamos que se familiaricen con el lenguaje de programación en assembly RISC-V[1] (*risk-five*), aplicando los conocimientos adquiridos en clases hasta el momento, en particular respecto a números de punto flotante, overflow, y otras propiedades de un computador básico, junto con observar y comprender el comportamiento de una memoria caché.

Enunciado

Parte 1: Bad cache

El script `cache.asm` disponible en el repositorio del curso, contiene código que al ejecutarlo en el emulador, conectado al *Data Cache Tool*, y con esta configurada para simular un cache de mapeo directo, con sustitución **LRU**, 1 *set* de bloques, 8 bloques y cada bloque de 8 words, obtiene un rendimiento horrible en términos de su *hit-rate*. Deberán modificar el programa de tal manera que no cambie el resultado de la ejecución (el valor de **a0** y **a1** al terminar la ejecución) ni los valores numéricos en la sección de datos, pero que mejore su rendimiento en términos de *hit-rate* en por lo menos 35%.

Deberán comentar todos los cambios que hicieron en el mismo script, e incluir un par de líneas con comentarios al comienzo del archivo, indicando por qué el script original tenía tan mal rendimiento.

3 puntos.

Evaluación parte 1:

El puntaje se asignará de la siguiente manera: 1 punto por mejorar el rendimiento del caché de manera sustantiva (al menos 35%) sin modificar el resultado final del programa. 1 punto por explicar los cambios realizados y porque el script pase la etapa del assembler y respete la convención de llamada. 1 puntos por explicar correctamente a que se debe el mal rendimiento del script original.

Parte 2: Regression

Para poder estrenar la flamante FPU de RISC-V que *claramente* no sabían que existía hasta hace poco, deberán escribir un programa que calcule una regresión lineal simple ($y = \alpha * x + \beta$) a partir de un arreglo

[1] `https://riscv.org/`

de pares de coordenadas, y entregue como resultado los valores de α, β y el factor r^2 para dicha regresión.

A modo de input, recibirán 2 labels, V y n, donde el primero corresponde a un arreglo de coordenadas `x,y`, y el segundo corresponde a la cantidad de puntos en el arreglo. Por lo tanto, la sección `.data` se deberá ver como sigue, respetando los labels que se indican (pueden agregar más si lo consideran necesario):

```
.data
    # --- No modificar labels ---
    V: .float 1.12,2.231, 5.67,6.32, 0.001,9.76, 10.124,22.765
    n: .word 4
    # --- End no modificar labels---
    # de aca para abajo van sus variables en memoria
```

En este caso, los puntos (x, y) son $(1.12, 2.231)$, $(5.67, 6.32)$, etc.

Al finalizar la ejecución, los valores de los registros `a0`, `a1` y `a2` deben ser los de α, β y r^2, respectivamente.

3 puntos.

Evaluación parte 2:

El código deberá estar adecuadamente comentado, de manera de facilitar la corrección y *debugging*, además de respetar las convenciones de llamada para los registros, especificadas en la segunda página del *green card*[2] de RISC-V. Por ejemplo, para guardar una variable en el registro `x5`, deberán hacerlo a través de `t0`. En su código nunca debiesen llamar a un registro a través de la notación `xN`, además de usar los registros de acuerdo con la descripción provista en el minicurso y la documentación.

```
# para sumar dos numeros, escribir
add a4, a3, a4
# NO HACER
add x14, x13, x14
```

RISC-V Calling Convention			
Register	ABI Name	Saver	Description
x0	zero	---	Hard-wired zero
x1	ra	Caller	Return address
x2	sp	Callee	Stack pointer
x3	gp	---	Global pointer
x4	tp	---	Thread pointer
x5-7	t0-2	Caller	Temporaries
x8	s0/fp	Callee	Saved register/frame pointer
x9	s1	Callee	Saved register
x10-11	a0-1	Caller	Function arguments/return values
x12-17	a2-7	Caller	Function arguments
x18-27	s2-11	Callee	Saved registers
x28-31	t3-t6	Caller	Temporaries
f0-7	ft0-7	Caller	FP temporaries
f8-9	fs0-1	Callee	FP saved registers
f10-11	fa0-1	Caller	FP arguments/return values
f12-17	fa2-7	Caller	FP arguments
f18-27	fs2-11	Callee	FP saved registers
f28-31	ft8-11	Caller	FP temporaries

Figure 1: Convención de llamada para registros. Free & Open RISC-V Reference Card, RISC-V Organization.

[2] https://inst.eecs.berkeley.edu/~cs61c/fa17/img/riscvcard.pdf

El desglose de puntos será el siguiente: 1 punto por pasar el assembler y respetar la convención de llamada. 1 punto por implementar correctamente una regresión lineal y entregar correctamente los valores de α y β para 4 tests. 1 punto por implementar correctamente el cálculo del coeficiente r^2 y entregar su valor correctamente en los mismos 4 tests.

Emulador

Si bien son libres de programar en el editor que prefieran e incluso compilar/emular en la herramienta que les sea más cómoda, la corrección será con el emulador RARS, disponible en el repositorio del curso, por lo que su tarea deberá pasar el *assembler* y ejecutar en dicho emulador. **No pasar la etapa del *assembler* en RARS implica 0 puntos en la sección de programación**.

Se recomienda encarecidamente evitar el uso de tildes (‘, ’, ˜ , etc.) u otros caracteres especiales en el código, ya que es probable que si hacen esto, el código se muestre con caracteres inválidos en el computador del ayudante por diferencias en cómo funciona el encoding/decoding de caracteres entre los diferentes sistemas operativos, y esto hará que su programa no pase el assembler.

Nota tarea

La nota se calculará de la siguiente manera:

$$Puntos + 1 = Nota\ tarea$$

La nota de la tarea se redondea a la decena.

Entrega

La fecha de entrega es el lunes 9 de mayo, hasta las 23:59 por Canvas. El formato a entregar será un archivo `.asm` por item, con sus respectivas soluciones en assembly RISC-V de la tarea. Se habilitará en Canvas un buzón para la tarea.