Universidad Católica San Pablo | Departamento de Ciencia de la Computación

Arquitectura de Computadores 2025-I (CCOMP3-1)
Laboratorio 03a: MIPS Básico

Yván Jesús Túpac Valdivia
Universidad Católica San Pablo, Arequipa – Perú

16 de abril de 2025

---

1 Objetivos

- Familiarizarse con la arquitectura de instrucciones MIPS y practicar con el emulador MARS/SPIM.
- Implementar y ejecutar programas MIPS secuenciales y observar cómo se ejecutan instrucciones MIPS básicas.
- Comprender e implementar programas MIPS para programación estructurada.

2 Contenido base

- Ensamblador: visión abstracta del *hardware*, instrucciones básicas MIPS, formatos de instrucciones, entrada y salida de datos.
- Diapositivas *Instructions Set Architecture (ISA) and MIPS*
- Libro base [Parhami, 2005].
- Emulador de máquina MIPS MARS/PC-SPIM: emulador, ensamblador de instrucciones MIPS.

3 Emulador de Instrucciones MIPS

3.1 MARS

Descargar, ejecutar y verificar el correcto funcionamiento del programa Emulador MARS (que es una aplicación en Java `.jar`). Documentación, gitgub con código fuente y un release de 2024 puede verse en: MARS MIPS Assembler and Runtime Simulator

4 MIPS básico

Ejecutar un programa simple dentro de MARS. El programa `Fibonacci.asm` escribe en consola la serie de Fibonacci hasta una cantidad de términos que se entra por el teclado.

- Este código lo que hace es utilizar varios conceptos de MIPS, que se verán en las siguientes clases, como:
  - Pseudo instrucciones.
  - Llamadas a servicios del sistema operativo (principalmente lectura e impresión por consola).
  - Instrucciones aritméticas.
  - Instrucciones de salto condicional e incondicional.
  - Variables en memoria para la impresión.

- Las llamadas a servicios del sistema operativo sirven para ejecutar la entrada y salida en consola, similar a cuando en C se hace `printf("Hello World");` o en C++ `cout << "Hello World";`.
- En MIPS el modo de entrada y salida en MIPS se realiza usando la instrucción de "llamada a servicios del sistema" **`syscall`**.
- Una llamada a servicios del sistema consiste en una llamada a subrutina y necesita un argumento entero en el registro `$v0` con valores entre 1 y 10. La Tabla 1 contiene las funciones asociadas con estas 10 posibilidades:

Tipo	$v0	Función	Argumentos	Resultados
Output	1	Imprimir registro entero	entero en $a0	Entero en consola
	2	Imprimir registro float	float en $f12	Float en consola
	3	Imprime registro double	double en $f12-$f13	Double en consola
	4	Imprimir string	puntero en $a0	string NULL-end en consola
Input	5	Leer entero		Entero leído en $v0
	6	Leer a registro float		Float leído en $f0
	7	Leer a registros double		Double leído en $f0-$f1
	8	Leer a string	Puntero en $a0, longitud en $a1	Cadena leída en puntero
Control	9	Reservar memoria	Núm de bytes en $a0	Puntero a bloque en $v0
	10	Salir de programa		Ejecución finalizada

Table 1: Parametrizaciones de syscall y su funcionamiento

- Coloque el código Fibonacci.asm en MARS, ensámblelo y verifique cómo se presentan las instrucciones ya ensambladas con respecto al código original de `Fibonacci.asm`. Algunas instrucciones cambiarán notablemente al ser ensambladas, lo que se debe al uso de pseudo instrucciones.
- Ejecute el programa y verifique el funcionamiento correcto, pruebe diversas cantidades de valores a generar.
- El simulador tiene opciones de ejecución paso a paso y con *breakpoints* con las que se puede apreciar cómo la ejecución de cada instrucción modifica el contenido de los registros de acuerdo a la lógica del programa.

4.1 Mapa de uso de memoria

1. Note la figura que muestra la organización de la memoria MIPS, existe un bloque denominado segmento de texto (o segmento de código) y otro denominado segmento de datos.
2. Revise los rangos de direcciones de memoria usadas en la figura 1, y compárelos con los segmentos de texto y datos vistos en el emulador.

4.2 Pseudo instrucciones

Una "Pseudo instrucción" es una instrucción que tiene la sintaxis de una instrucción MIPS, pero al ensamblarse se convierte en una o varias instrucciones reales MIPS cuyo funcionamiento es el indicado por la pseudo instrucción. Observe y explique por qué el código que se presenta en el archivo `Fibonacci.asm` es diferente al del segmento de código ya ensamblado que muestra el emulador.

4.3 Ejecución de Fibonacci

- Ejecute `Fibonacci.asm` en el emulador.
- Observe y experimente el funcionamiento del programa. Observe la ventana de datos notando cómo se van modificando los valores del archivo de registros hasta encontrar las respuestas deseadas.
- Revise cada una de las instrucciones y directivas de compilación que aparecen en el programa. Identifique las pseudoinstrucciones en este programa.

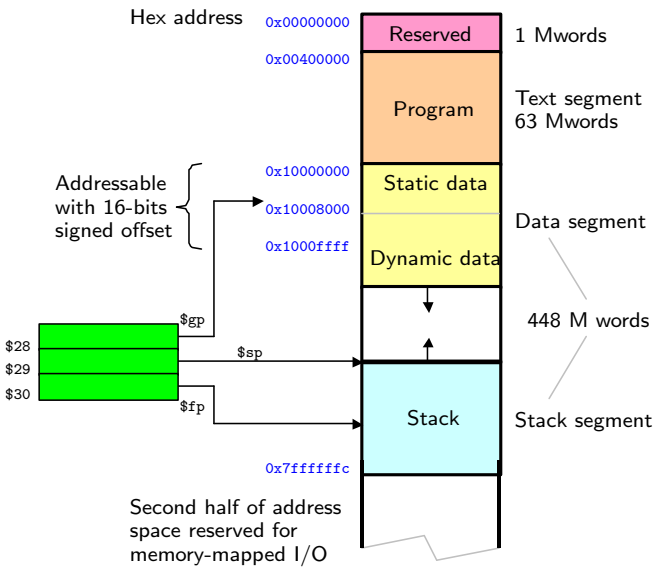

Figure 1: Mapa de memoria de un sistema MIPS

- Haga un resumen indicando sus observaciones, incluya un diagrama de flujo y haga el pseudo código de este programita.
- Respuestas a estas tres preguntas deben presentarse en un informe en la siguiente clase/sesión práctica.

5 MIPS para bucles y saltos condicionales

5.1 Programa Upcase

- El programa que está en el archivo `Upcase.asm`, tiene como objetivo revisar una cadena con una frase escrita y convierte todas las letras iniciales de la cadena a mayúsculas, sumando un valor al código ASCII correspondfiente, pero verificando antes si son letras y están en minúscula.
- A partir de este código haga lo siguiente:

  1. Verifique el correcto funcionamiento del ensamblado y ejecución de la conversión a mayúsculas, si detecta casos en que ocurre error, explicar el error, hacer la correción y crear una versión `Upcase_2.0.asm`
  2. Elabore un diagrama de flujo y pseudo algoritmo del código analizado, si detectó y corrigió algún error, también resaltarlo en el diagrama y pseudo código.

References

[Parhami, 2005] Parhami, B. (2005). *Computer Architecture: From Microprocessors to Supercomputers*. The Oxford Series in Electrical and Computer Engineering. OUP USA.

6 Anexo - Código Fibonacci.asm

```
## Fibonacci.asm

# SECCION DE INSTRUCCIONES (.text)
.text
.globl __start

__start:
la $a0, Fiboprt
li $v0, 4
syscall
li $v0, 5
syscall
addi $t8,$v0,0          # valor de teclado en $t8
li $t0,0
li $t1,1

la $a0,Fibost1
li $v0,4
syscall                 # "La serie Fibonacci de "
addi $a0,$t8,0
li $v0,1
syscall                 # n
la $a0,Fibost2
li $v0,4
syscall                 # " terminos es: "
li $a0,1
li $v0,1
syscall                 # 1, ...
la $a0,coma
li $v0,4
syscall
      li   $t4,2
      beq  $t8,$0,fin
      bltz $t8,fin
loop: add  $t2,$t0,$t1  # fibonacci
      addi $a0,$t2,0
      li $v0,1
      syscall
      beq $t4,$t8,fin
      la $a0,coma
      li $v0,4
      syscall
      addi $t4,$t4,1
      addi $t0,$t1,0
      addi $t1,$t2,0

      j loop

fin:
      la $a0,endl
      li $v0,4
      syscall
      li $v0,10
      syscall

# SECCION DE VARIABLES EN MEMORIA (.data)
.data
Fiboprt: .asciiz "Ingresar␣n:"
```

```
Fibost1: .asciiz "La␣serie␣Fibonacci␣de␣"
Fibost2: .asciiz "␣terminos␣es:␣"
coma:    .asciiz ",␣"
endl:    .asciiz "\n"
```