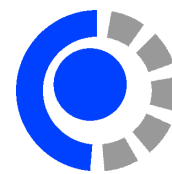




## Arquitecturas y Organización de Computadoras I 2° Cuatrimestre 2020

### TP N° 5 – Introducción a la Programación en lenguaje ensamblador MIPS con MIPSX



Facultad de  
Informática

**Objetivo:** Comprender la estructura de un programa en lenguaje ensamblador MIPS básico. Introducción a la interfaz `mipsx` y al conjunto de instrucción MIPS.

#### Recursos y Bibliografía:

Arq. MIPS Vol I, II and III.

Programa `mipsx` desarrollado por la cátedra.

### Introducción a `mipsx`

El programa `mipsx`, ha utilizar en este trabajo práctico, es una interfaz gráfica para desarrollar programas en lenguaje ensamblador MIPS. Es desarrollado por la cátedra y trabaja en conjunto con sistemas MIPS emulados y reales.

`mipsx` permite ensamblar y vincular los programas desarrollados. También ejecutar, y al mismo tiempo analizar, los programas compilados a través del debugger `gdb`. A partir de estas características, es posible realizar todo el proceso de desarrollo y verificación de programas en lenguaje ensamblador con una única herramienta, mientras que los programas pueden ser ejecutados y analizados en diferentes sistemas MIPS.

A continuación se detallan las partes fundamentales de esta interfaz.

**Editor del Programa**

```
# Hello World en MIPS
#
.data
memoria:
.word 0xabcd1234
.asciz "hola mundo"

.text
.global main
.global __start

main:
__start:
li    $t1, 1
li    $t3, 1
add   $t0, $t1, $t3

move $a0, $0      # exit status as 0
li $v0, 4001
syscall
```

**Registros**

Register	Value
zero	00000000
at	00000000
v0	00000000
v1	00000000
a0	00000000
a1	00000000
a2	00000000
a3	00000000
t0	00000000
t1	00000001
t2	00000000
t3	00000001
t4	00000000
t5	00000000
t6	00000000
t7	00000000
s0	00000000
s1	00000000
s2	00000000
s3	00000000
s4	00000000
s5	00000000
s6	00000000
s7	00000000
t8	00000000
t9	00000000
k0	00000000
k1	00000000
gp	00000000
sp	7ff75568
s8	00000000
ra	00000000
status	00000000
lo	00000000
hi	00000000
badvaddr	00000000
cause	00000000
pc	00000000
fcsr	00000000
fir	00000000
restart	00000000

**Programa en Assembler y Programa Binario Decodificado (disassemble)**

```
0x004000b4 <+4>: li t3,1
0x004000b8 <+8>: add t0,t1,t3
0x004000bc <+12>: move a0,zero
0x004000c0 <+16>: li v0,4001
=> 0x004000c4 <+20>: syscall
0x004000c8 <+24>: nop
0x004000cc <+28>: nop
End of assembler dump.
```

**Mensajes de Depuracion**

```
Ejecucion FINALIZADA

No stack.
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobs
pec ... or kill -l [sigspec]
```

**Memoria - Segmento de datos (debe existir la etiqueta "memoria") - Segmento de texto - Pila**

Address	Value
0x4100d0	0xabcd1234
0x4100e0	0x686f6c61
0x4100f0	0x206d756e
0x410100	0x646f0000
0x410110	0x00000001
0x410120	0x00000007
0x410130	0x0000000f
0x410140	0x00000001
0x410150	0x64000000
0x410160	0x004000b0
0x410170	0x00000000
0x410180	0x00000000
0x410190	0x00000000
0x4101a0	0x00000000
0x4101b0	0x00000000
0x4101c0	0x00000000
0x4101d0	0x00000000
0x4101e0	0x00000000
0x4101f0	0x00000000
0x410200	0x00000000
0x410210	0x00000000
0x410220	0x00000000
0x410230	0x00000000
0x410240	0x00000000
0x410250	0x00000000
0x410260	0x00000000
0x410270	0x00000000
0x410280	0x00000000
0x410290	0x00000000
0x4102a0	0x00000000
0x4102b0	0x00000000
0x4102c0	0x00000000
0x4102d0	0x00000000
0x4102e0	0x00000000
0x4102f0	0x00000000
0x410300	0x00000000
0x410310	0x00000000
0x410320	0x00000000
0x410330	0x00000000
0x410340	0x00000000
0x410350	0x00000000
0x410360	0x00000000
0x410370	0x00000000
0x410380	0x00000000
0x410390	0x00000000
0x4103a0	0x00000000
0x4103b0	0x00000000
0x4103c0	0x00000000
0x4103d0	0x00000000
0x4103e0	0x00000000
0x4103f0	0x00000000
0x410400	0x00000000
0x410410	0x00000000
0x410420	0x00000000
0x410430	0x00000000
0x410440	0x00000000
0x410450	0x00000000
0x410460	0x00000000
0x410470	0x00000000
0x410480	0x00000000
0x410490	0x00000000
0x4104a0	0x00000000
0x4104b0	0x00000000
0x4104c0	0x00000000
0x4104d0	0x00000000
0x4104e0	0x00000000
0x4104f0	0x00000000
0x410500	0x00000000
0x410510	0x00000000
0x410520	0x00000000
0x410530	0x00000000
0x410540	0x00000000
0x410550	0x00000000
0x410560	0x00000000
0x410570	0x00000000
0x410580	0x00000000
0x410590	0x00000000
0x4105a0	0x00000000
0x4105b0	0x00000000
0x4105c0	0x00000000
0x4105d0	0x00000000
0x4105e0	0x00000000
0x4105f0	0x00000000
0x410600	0x00000000
0x410610	0x00000000
0x410620	0x00000000
0x410630	0x00000000
0x410640	0x00000000
0x410650	0x00000000
0x410660	0x00000000
0x410670	0x00000000
0x410680	0x00000000
0x410690	0x00000000
0x4106a0	0x00000000
0x4106b0	0x00000000
0x4106c0	0x00000000
0x4106d0	0x00000000
0x4106e0	0x00000000
0x4106f0	0x00000000
0x410700	0x00000000
0x410710	0x00000000
0x410720	0x00000000
0x410730	0x00000000
0x410740	0x00000000
0x410750	0x00000000
0x410760	0x00000000
0x410770	0x00000000
0x410780	0x00000000
0x410790	0x00000000
0x4107a0	0x00000000
0x4107b0	0x00000000
0x4107c0	0x00000000
0x4107d0	0x00000000
0x4107e0	0x00000000
0x4107f0	0x00000000
0x410800	0x00000000
0x410810	0x00000000
0x410820	0x00000000
0x410830	0x00000000
0x410840	0x00000000
0x410850	0x00000000
0x410860	0x00000000
0x410870	0x00000000
0x410880	0x00000000
0x410890	0x00000000
0x4108a0	0x00000000
0x4108b0	0x00000000
0x4108c0	0x00000000
0x4108d0	0x00000000
0x4108e0	0x00000000
0x4108f0	0x00000000
0x410900	0x00000000
0x410910	0x00000000
0x410920	0x00000000
0x410930	0x00000000
0x410940	0x00000000
0x410950	0x00000000
0x410960	0x00000000
0x410970	0x00000000
0x410980	0x00000000
0x410990	0x00000000
0x4109a0	0x00000000
0x4109b0	0x00000000
0x4109c0	0x00000000
0x4109d0	0x00000000
0x4109e0	0x00000000
0x4109f0	0x00000000
0x410a00	0x00000000
0x410a10	0x00000000
0x410a20	0x00000000
0x410a30	0x00000000
0x410a40	0x00000000
0x410a50	0x00000000
0x410a60	0x00000000
0x410a70	0x00000000
0x410a80	0x00000000
0x410a90	0x00000000
0x410aa0	0x00000000
0x410ab0	0x00000000
0x410ac0	0x00000000
0x410ad0	0x00000000
0x410ae0	0x00000000
0x410af0	0x00000000
0x410b00	0x00000000
0x410b10	0x00000000
0x410b20	0x00000000
0x410b30	0x00000000
0x410b40	0x00000000
0x410b50	0x00000000
0x410b60	0x00000000
0x410b70	0x00000000
0x410b80	0x00000000
0x410b90	0x00000000
0x410ba0	0x00000000
0x410bb0	0x00000000
0x410bc0	0x00000000
0x410bd0	0x00000000
0x410be0	0x00000000
0x410bf0	0x00000000
0x410c00	0x00000000
0x410c10	0x00000000
0x410c20	0x00000000
0x410c30	0x00000000
0x410c40	0x00000000
0x410c50	0x00000000
0x410c60	0x00000000
0x410c70	0x00000000
0x410c80	0x00000000
0x410c90	0x00000000
0x410ca0	0x00000000
0x410cb0	0x00000000
0x410cc0	0x00000000
0x410cd0	0x00000000
0x410ce0	0x00000000
0x410cf0	0x00000000
0x410d00	0x00000000
0x410d10	0x00000000
0x410d20	0x00000000
0x410d30	0x00000000
0x410d40	0x00000000
0x410d50	0x00000000
0x410d60	0x00000000
0x410d70	0x00000000
0x410d80	0x00000000
0x410d90	0x00000000
0x410da0	0x00000000
0x410db0	0x00000000
0x410dc0	0x00000000
0x410dd0	0x00000000
0x410de0	0x00000000
0x410df0	0x00000000
0x410e00	0x00000000
0x410e10	0x00000000
0x410e20	0x00000000
0x410e30	0x00000000
0x410e40	0x00000000
0x410e50	0x00000000
0x410e60	0x00000000
0x410e70	0x00000000
0x410e80	0x00000000
0x410e90	0x00000000
0x410ea0	0x00000000
0x410eb0	0x00000000
0x410ec0	0x00000000
0x410ed0	0x00000000
0x410ee0	0x00000000
0x410ef0	0x00000000
0x410f00	0x00000000
0x410f10	0x00000000
0x410f20	0x00000000
0x410f30	0x00000000
0x410f40	0x00000000
0x410f50	0x00000000
0x410f60	0x00000000
0x410f70	0x00000000
0x410f80	0x00000000
0x410f90	0x00000000
0x410fa0	0x00000000
0x410fb0	0x00000000
0x410fc0	0x00000000
0x410fd0	0x00000000
0x410fe0	0x00000000
0x410ff0	0x00000000
0x411000	0x00000000
0x411010	0x00000000
0x411020	0x00000000
0x411030	0x00000000
0x411040	0x00000000
0x411050	0x00000000
0x411060	0x00000000
0x411070	0x00000000
0x411080	0x00000000
0x411090	0x00000000
0x4110a0	0x00000000
0x4110b0	0x00000000
0x4110c0	0x00000000
0x4110d0	0x00000000
0x4110e0	0x00000000
0x4110f0	0x00000000
0x411100	0x00000000
0x411110	0x00000000
0x411120	0x00000000
0x411130	0x00000000
0x411140	0x00000000
0x411150	0x00000000
0x411160	0x00000000
0x411170	0x00000000
0x411180	0x00000000
0x411190	0x00000000
0x4111a0	0x00000000
0x4111b0	0x00000000
0x4111c0	0x00000000
0x4111d0	0x00000000
0x4111e0	0x00000000
0x4111f0	0x00000000
0x411200	0x00000000
0x411210	0x00000000
0x411220	0x00000000
0x411230	0x00000000
0x411240	0x00000000
0x411250	0x00000000
0x411260	0x00000000
0x411270	0x00000000
0x411280	0x00000000
0x411290	0x00000000
0x4112a0	0x00000000
0x4112b0	0x00000000
0x4112c0	0x00000000
0x4112d0	0x00000000
0x4112e0	0x00000000
0x4112f0	0x00000000
0x411300	0x00000000
0x411310	0x00000000
0x411320	0x00000000
0x411330	0x00000000
0x411340	0x00000000
0x411350	0x00000000
0x411360	0x00000000
0x411370	0x00000000
0x411380	0x00000000
0x411390	0x00000000
0x4113a0	0x00000000
0x4113b0	0x00000000
0x4113c0	0x00000000
0x4113d0	0x00000000
0x4113e0	0x00000000
0x4113f0	0x00000000
0x411400	0x00000000
0x411410	0x00000000
0x411420	0x00000000
0x411430	0x00000000
0x411440	0x00000000
0x411450	0x00000000
0x411460	0x000000

El programa presenta cinco paneles:

**Editor del Programa:** es el editor ha utilizar para el desarrollo de los programas en lenguaje ensamblador.

**Registros:** presenta el contenido de los registros del procesador (CPU).

**Programa en assembler y Programa Binario Decodificado:** muestra el listado del programa con sus respectivos números de línea. Permite seguir la traza de las instrucciones ejecutadas del programa siendo procesado por gdb.

También presenta un “disassemble” del programa en memoria, muy útil para reconocer pseudoinstrucciones y comparar con el programa original.

**Memoria:** permite visualizar el contenido de la memoria de la máquina MIPS. En particular, los segmentos del programa en memoria, de datos, de texto, y pila.

**Mensajes de depuración:** muestra información del estado del ensamblaje y vinculación, carga, y mensajes del programa siendo ejecutado por gdb, y la salida estándar.

El panel superior contiene los botones que controlan a `mipsx`:

**Run :** Ejecuta el programa siendo analizado hasta su finalización.

**Next :** Ejecuta la siguiente instrucción del programa en ejecución.

**Breakpoint :** Define o elimina breakpoints (NO IMPLEMENTADO).

**Compilar y Cargar :** copia, ensambla y vincula, en el sistema MIPS remoto, el programa siendo editado. Carga el programa ejecutable con gdb e inicia la ejecución de la primera instrucción del programa si no se presentaron errores.

**NOTA:** Para ejecutar `mipsx` debe contar con un usuario de red en los laboratorios de Pcs Linux. Si todavía no tiene una cuenta de usuario de red puede solicitar una a la cátedra. La cuenta será de utilidad para el resto de las materias en la carrera.

**1.** Ejecutar el programa `mipsx`: En una terminal Linux debe conectarse remotamente a una computadora de la facultad utilizando el siguiente comando, y reemplazando `USUARIO` por el suyo. Tenga en cuenta que al momento de ingresar la contraseña, no se hace eco de las teclas presionadas (no se muestra nada, ni siquiera asteriscos). Para usuarios de windows, ver el anexo al final del documento.

```
$ ssh -X -p 12222 USUARIO@debianmips.fi.uncoma.edu.ar
```

Una vez conectado, para ejecutar la herramienta `mipsX` debe ingresar el siguiente comando:

```
$ /export/home/extras/mipsx/mipsx.sh
```

**2.** Desarrollar el siguiente programa utilizando `mipsx`

**IMPORTANTE:** No utilice la función COPIAR y PEGAR para transferir el programa a `mipsx`. Esta acción suele “copiar” caracteres no-visibles, que puede resultar en un programa en `mipsx` que no está apto para ser ensamblado.

```
# Hello World en MIPS
#
        .data                # Zona de datos
memoria:
        .word 2, 3
        .word 0xabcd1234
        .ascii "hola mundo"
```

```

        .text           # Zona del programa
        .global main    # La etiqueta main debe ser global
        .global __start # La etiqueta __start debe ser global
__start:
main:
    la      $t0, memoria
    lw      $t1, 0($t0)
    lw      $t2, 4($t0)
    add     $t3, $t1, $t2
    sw      $t3, 8($t0)

# finaliza el programa solicitando un `exit` al sistema Linux
    move $a0, $0      # exit status as 0
    li $v0, 4001      # __NR_exit include/asm-mips/unistd.h
    syscall

```

Las palabras que comienzan con un punto: `.data`, `.word`, `.text` y `.globl` son directivas al compilador. Se utilizan para dar ordenes especiales acerca de cómo debe generarse el código. Los significados de cada una de las directivas utilizadas en este programa son:

- **.data**: los siguientes elementos han de situarse en el segmento de datos.
- **.word**, **.byte**, **.float**, **.ascii**, **etc.**: especifican el tipo de los datos que se detallan a continuación de la directiva.
- **.text**: indica al ensamblador que los siguientes elementos han de situarse en el segmento de texto (programas) del usuario.
- **.globl**: declara la etiqueta que le sigue como global, de forma tal que pueda ser referenciada desde el código del sistema que inicia nuestro programa.

**3. Compilar y Cargar el programa.** Verificar en el panel de Depuración que no se han presentado errores al compilar el programa.

Si el programa se cargó con `gdb` sin errores puede analizar el contenido de los diferentes paneles de información. Conteste :

1. ¿Cuál es el valor de registro del procesador `$pc`?
2. ¿Cuál es el valor del registro `$0` (`$zero`)?
3. Analice con el registro `$pc` y los diferentes paneles cuál es la próxima instrucción que se ejecutará en la CPU MIPS. ¿Puede indicar la línea de su programa que corresponde a la próxima instrucción a ejecutar por la CPU? (Indicar la línea del programa que corresponde a esta instrucción y la instrucción en lenguaje ensamblador).

**4. Ejecución paso a paso:** Compilar y cargar el programa nuevamente. Luego, ejecute el programa paso a paso utilizando la orden `Next`. Cada vez que demos la orden de ejecutar un paso, se ejecutará una única instrucción y se detendrá la ejecución del programa, para que podamos observar el efecto producido por la instrucción. **Tener en cuenta que al Compilar y Cargar el programa, `mipsx` ya ejecuta la primera instrucción del programa y detiene la ejecución.** Contestar:

1. ¿Cuál es la dirección en memoria de la primera instrucción del programa?
2. ¿Cuál es la dirección en memoria de la última instrucción del programa?
3. Con la información anterior responda: ¿Cuántos bytes ocupa el segmento de código en memoria? (de instrucciones).

**5. Ejecución del programa.** Cargue nuevamente y Ejecute el programa completo utilizando la orden Run. La máquina MIPS ejecutará el programa por completo hasta su finalización. Cuando la máquina MIPS termine la ejecución se indicará en el panel de Depuración con la leyenda “Ejecución FINALIZADA”.  
Conteste:

1. ¿Cuál es el valor del registro t1, t2 y t3 al finalizar el programa?
2. Explique el funcionamiento del programa.
3. Explique qué sucedió con la palabra de texto “hola” que se encontraba en memoria.

**6.** Copie el programa 4 del TP04 para calcular el enésimo elemento de la sucesión fibonacci, y compruebe el correcto funcionamiento del programa desarrollado. No olvide agregar el siguiente segmento de código al final del programa para que el proceso pueda terminar correctamente, además de la declaración de `__start`. Se recomienda utilizar la ejecución paso a paso para detectar los errores.

```
# finaliza el programa solicitando un `exit` al sistema Linux
    move $a0, $0          # exit status as 0
    li $v0, 4001          # __NR_exit include/asm-mips/unistd.h
    syscall
```

**7.** Sabiendo que X e Y son dos variables declaradas como enteros, y utilizando solo las instrucciones de salto beq, bne y j, y slt para comparar entre registros, cree fragmentos de código máquina que implementen las siguientes estructuras de control:

1. if (X==Y) { /\*código\*/ }
2. if (X<Y) { /\*código\*/ } else { /\*código2\*/ }
3. while (X!=11) { /\*código\*/ }
4. do { /\*código\*/ } while (X!=0)
5. for (X=0; X<Y; X++) { /\*código\*/ }

**Ejemplo:**

La estructura de control `while (true) { /*código*/ }` puede ser implementada con el siguiente código MIPS:

```
salto1:
    #código
j salto1
```

## Anexo

Para aquellas personas que necesiten utilizar el programa mipsx desde Windows e internet pueden instalar putty y xming y conectarse.

Instrucciones :

1. Bajar putty : <http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe>
2. Instalar xming : <http://sourceforge.net/projects/xming/>
3. En las opciones de instalacion marcar "No instalar el cliente ssh"
4. Una vez instalado se debe tener "en ejecución" el programa xming (debería estar el ícono en la barra de programas de ejecución). Con xming en ejecución ejecutan putty que bajaron previamente.
5. Este paso es importantísimo. Si no se tiene en EJECUCIÓN el programa xming, las instrucciones que siguen debajo no funcionan:
  - 5.1. Colocan como nombre de host : `debianmips.fi.uncoma.edu.ar`
  - 5.2. PORT : 12222
  - 5.3. En las opciones de SSH -> X11 : habilitan "X11 forwarding".
  - 5.4. Le dan click a "open". En ese punto putty se conecta y les pide el nombre de usuario y la clave. Utilizan su usuario y clave de los laboratorios (la clave no aparece cuando la escriben).
  - 5.5. Una vez que ingresan pueden ejecutar mipsx con el comando:  
`/export/home/extras/mipsx/mipsx.sh`