Resumen Practico - Mod 2

Atascos

Atasco estructural

Debido a que EX puede tener multiples instrucciones en simultaneo. Puede ocurrir que dos o mas de ellas terminen al mismo tiempo y en el proximo ciclo quieran acceder a la etapa de MEM.

- La solucion es darle prioridad al que entro primero a **EXE.**

Atascos WAR y WAW

War: Esta intentado escribir algo que todavia no fue leido

- Una instrucción puede sobrepasar a una instrucción anterior, queriendo escribir un registro pendiente de lectura (WAR) o escritura (WAW)

- El simulador produce atascos cuando detecta una situación potencial (**puede que realmente no suceda**) de dependencia WAR o WAW.

Atasco RAW:

- **Importante:** En casos donde hay que hacer LD, el valor esta disponible a partir de MEM(si hay **forwarding** activado)o en WB(si no hay forwarding activado). En operaciones aritmeticas aparece en etapa EXE (**forwarding activado)** o WB(forwarding no activado).
- BRANCH TAKEN STALL: Se sigue con el pipeline cuando debe saltar a una instruccion anterior
  - **Solucion:** Predecir si se va a saltar o no.
  - Posible problema **BRANCH MISTAKEN STALL,** lo cual significa que se le erro a la prediccion(esta prediccion esta en el **branch target buffer)**.

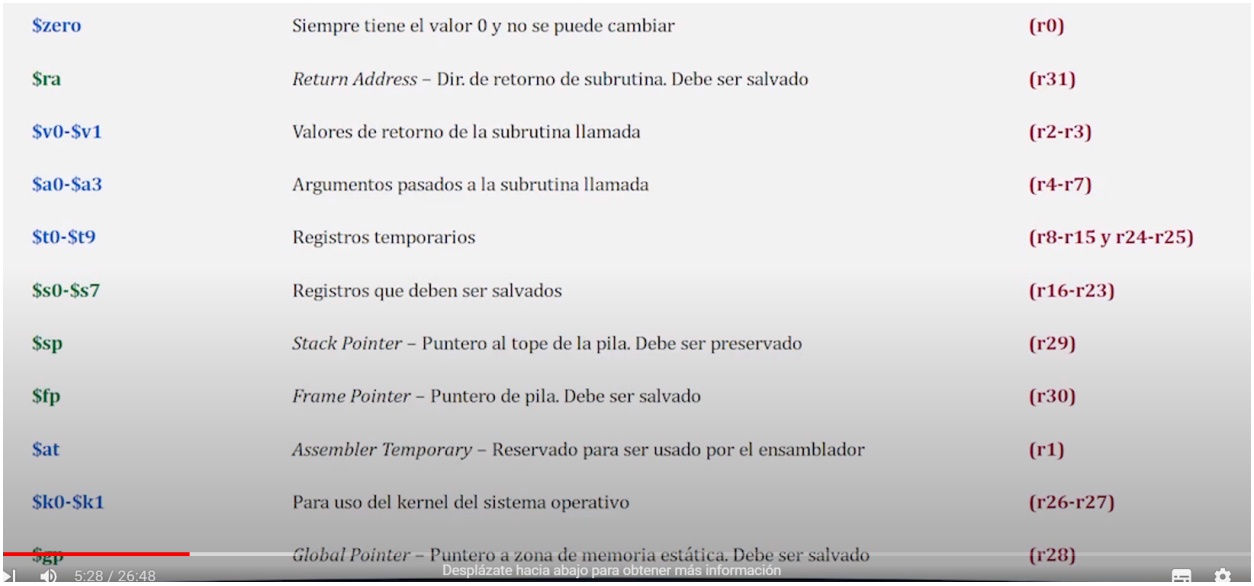
Funcionamiento del **Branch target buffer:**

- Inicia en **no saltes.**
- Si le erra a la prediccion actualiza a **salta.** Este cambio del BTB se cuenta como un **atasco.**
    - Si predice **salto** y no habia que saltar cambia BTB.

DELAY SLOT: Siempre ejecuta la proxima instruccion.

Subrutinas

Es conveniente olvidar los Rn y acordarse de los nuevos nombres.



- Unicamente guardo (*los guardo antes de la subrutina)* los **$Sn** que voy a usar en la subrutina.

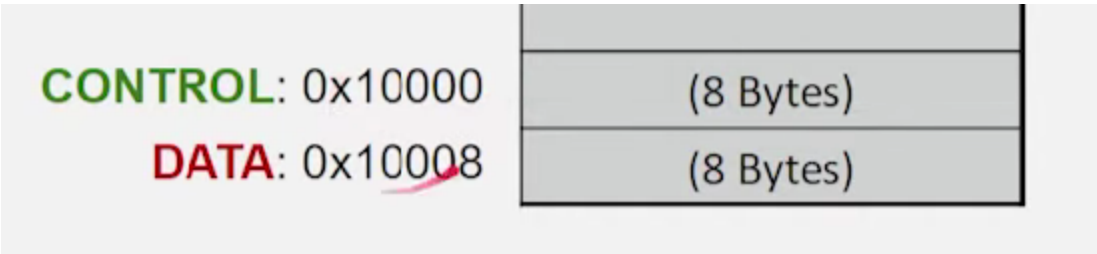
Inicializaciones

- **Inicializar argumentos con offset:** `$daddi $a0, $0, nombreCadena` .
- **Inicializar Stack Pointer:** `$sp, $0, 0x400`

Datos Importantes

`LBU $a0, 0($s0)` : Offset del *unsigned byte* que hay en $s0.

Entrada / Salida



Operamos como con cualquier celda de memoria

→ Para mandar datos a **CONTROL O DATA** usamos `LD $s1, DATA($0)` donde DATA/CONTROL son las direcciones **predefinidas** para ambos.

Impresion Alfanumerica

Si se quiere **imprimir** un string (**NO** un caracter, un string completo!)

DATA -> Dirección del string
CONTROL -> El valor *4*

Si se quiere **imprimir** un número

DATA -> El dato
CONTROL ->

- *1* -> Imprime un **entero sin signo**
- *2* -> Imprime un **entero con signo**
- *3* -> Imprime un **flotante**

Si se quiere **limpiar la pantalla**

CONTROL -> El valor *6*

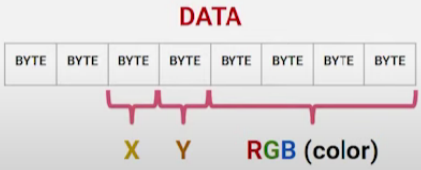
```
.data
CONTROL: .word 0x10000
DATA: .word 0x10008

.code
LD $s0, CONTROL ($zero) ; $s0 = CONTROL
LD $s1, DATA ($zero) ; $s1 = DATA

DADDI $t0, $zero, -85
SD $t0, 0 ($s1) ; Mando el dato a DATA

DADDI $t0, $zero, 2
SD $t0, 0 ($s0) ; CONTROL = 2

DADDI $t0, $zero, 6
SD $t0, 0 ($s0) ; CONTROL = 6 (limpia)
HALT
```

Impresion en Pantalla

Pantalla **gráfica**

Si se quiere **pintar un píxel**

DATA -> Color y coordenadas
CONTROL -> El valor *5*



Si se quiere **limpiar la pantalla**

CONTROL -> El valor *7*

```
.data
PIXEL: .byte 0, 185, 135, 0, 23, 10, 0, 0
CONTROL: .word 0x10000
DATA: .word 0x10008

.code
LD $s0, CONTROL ($zero) ; $s0 = CONTROL
LD $s1, DATA ($zero) ; $s1 = DATA

LD $t0, PIXEL ($zero)
SD $t0, 0 ($s1) ; Mando el dato a DATA

DADDI $t0, $zero, 5
SD $t0, 0 ($s0) ; CONTROL = 5

DADDI $t0, $zero, 7
SD $t0, 0 ($s0) ; CONTROL = 7 (limpia)
HALT
```

Lectura de teclado (INTS)

Podemos leer un número o un caracter:

Si se quiere **leer un número (entero o flotante)**

CONTROL -> El valor *8*
DATA ->

- **Muestra** el caracter presionado
- Termina de leer cuando presiona *Enter*
- Si el dato ingresado no es un número se guarda 0. Tomar el valor (**Hexadecimal**) con LD o L.D desde **DATA**

Si se quiere **leer un caracter**

CONTROL -> El valor *9*
DATA ->

- **NO** muestra el caracter presionado
- No espera al *Enter*
- Tomar el valor (**ASCII**) con LBU desde **DATA**

```
.data
CONTROL: .word 0x10000
DATA: .word 0x10008
NUM: .double 0.0
CARACTER: .byte 0

.code
LWU $s0, CONTROL ($zero)  ; $s0 = CONTROL
LWU $s1, DATA ($zero)     ; $s1 = DATA

DADDI $t0, $zero, 8
SD $t0, 0 ($s0) ; CONTROL = 8
L.D f1, 0 ($s1) ; Tomo número en f1
S.D f1, NUM ($zero) ; Guardo en variable

DADDI $t1, $zero, 9
SD $t1, 0 ($s0) ; CONTROL = 9
LBU $t1, 0 ($s1) ; Tomo caracter en $t1
SB $t1, CARACTER ($zero) ; Guardo en variable
HALT
```