El manual básico de CoreWar

Aprende Redcode DAT cero.

Fernando Méndez Torrubiano

21 de octubre de 2023

Índice

1. Introducción al CoreWar.

1.1. Historia

CoreWar es un juego de programación en donde combaten entre sí programas escritos en un lenguaje similar al ensamblador con el objetivo de ocupar toda la memoria de la máquina eliminando así a los oponentes.
El primer sistema de este tipo se denominó **Redcode** así como el lenguaje empleado.

1.2. Objetivo

Sobrescribir el programa del rival y/o hacer que ejecute una instrucción ilegal (**DAT**).

1.3. Conceptos básicos

- **Ejecución de instrucciones:**
  Cada instrucción **Redcode** ocupa exactamente una posición de memoria y tarda exactamente un ciclo de reloj para ejecutarse.
  Sin embargo, la velocidad a la que un proceso ejecuta instrucciones depende del número de otros procesos en la cola, ya que el tiempo de procesamiento se comparte por igual.

- **Memoria circular:**
  Cada celda de memoria puede estar ocupada por una sola instrucción. El espacio de memoria (o núcleo) es de tamaño finito (*CORESIZE)*, pero sólo se utiliza el direccionamiento relativo, es decir, la dirección 0 siempre hace referencia a la instrucción que se ejecuta actualmente, dirección 1 a la instrucción después de ella, y así sucesivamente.
  Por tanto, a pesar de que la memoria tenga un tamaño finito, nunca tendrá un final, pues la última y primera posición, son contiguas y cambian a lo largo de la ejecución del código.

- **Multiprocesamiento de bajo nivel:**
  En lugar de un solo puntero de instrucción, el simulador de **Redcode** tiene una cola de procesos para cada programa que contiene un número variable de punteros de instrucción que el simulador recorre.
  Cada programa comienza con un solo proceso, pero se pueden agregar nuevos procesos a la cola mediante la instrucción *SPL*. Un proceso muere cuando ejecuta una instrucción *DAT* o realiza una división por cero. Un programa se considera muerto cuando no tiene más procesos a la izquierda.

- **Etiquetas:**
  Para facilitar el uso de las direcciones de memoria, el **Redcode** moderno, permite el uso de etiquetas antes de una instrucción. De tal forma que, podremos saltar a la dirección de esa etiqueta en cualquier momento.

2. Instrucciones básicas.

Conceptos básicos

- `DIRECCIONAMIENTO DIRECTO`: **Las direcciones de memoria se escriben con '$' (o directamente con un número entero, sin nada más)**. Con esto, se da una dirección de memoria, la cual contiene un valor.
- `INMEDIATOS`: **Los inmediatos se indican con una '#' delante del número**, esto representa el valor del entero dado.
- `COMENTARIOS`: Los comentarios se escriben con un ';'.
- `ETIQUETAS`: Se pueden definir etiquetas de salto escribiendo el nombre de la etiqueta y ':' antes de la instrucción a la que se quiere saltar, por ejemplo: ***miEtiqueta: MOV 0, 1***

2.1. DAT

Descripción de la instrucción

Mata el proceso.

Ejemplo de uso de la instrucción

- **DAT #0, #0** ;Casi imprescindible si queremos ganar, aunque se puede volver en nuestra contra.

2.2. MOV

Descripción de la instrucción

Copia el dato de una dirección a otra.

Ejemplos de uso de la instrucción

- **MOV dir1, dir2** ;Copia el contenido de la dirección 'dir1' en la 'dir2'.
- **MOV inm, dir** ;Copia el valor del inmediato 'inm' en la dirección 'dir'.

2.3. ADD/SUB

Descripción de la instrucción

Suma / Resta.

Ejemplos de uso de la instrucción

- **ADD dir1, dir2** ;Suma el contenido de la dirección 'dir1' al de la 'dir2'.
- **ADD inm, dir** ;Suma el valor 'inm' al contenido de la dirección 'dir'.
- **SUB dir1, dir2** ;Resta el contenido de la dirección 'dir1' al de la 'dir2'.
- **SUB inm, dir** ;Resta el valor 'inm' al contenido de la dirección 'dir'.

2.4. JMP

Descripción de la instrucción

Salto incondicional.

Ejemplos de uso de la instrucción

- **JMP dir** ;Salta a la dirección 'dir'.
- **JMP etiqueta** ;Salta a la dirección de la etiqueta.

3. Instrucciones avanzadas.

Conceptos avanzados

- `DIRECCIONAMIENTO DIRECTO`: **Los direccionamientos directos a memoria se escriben con '$' (o directamente con un número entero, sin nada más)**.
  Con esto, se da una dirección de memoria que contiene un valor.
- `DIRECCIONAMIENTO INDIRECTO`: **Los direccionamientos indirectos a memoria se escriben con '@' (o directamente con un número entero, sin nada más)**.
  Con esto, se da una dirección de memoria que contiene otra dirección de memoria, la cual contiene, ahora sí, un valor.
- `DIRECCIONAMIENTO INDIRECTO CON PREDECREMENTO`: **Los direccionamientos indirectos con pre-decremento se escriben con '<'**.
  En este modo, primero se decrementa el valor en la ubicación de memoria y luego se utiliza ese valor para acceder a la memoria.
- `DIRECCIONAMIENTO INDIRECTO CON POSTINCREMENTO`: **Los direccionamientos indirectos con post-incremento se escriben con '>'**.
  En este modo, primero se utiliza el valor en la ubicación de memoria para acceder a la memoria, y luego se incrementa el valor en esa ubicación.
- `CONSTANTES`: **Existen constantes, como el tamaño del núcleo (`CORESIZE`) o el número de programas en ejecución (`WARRIORS`).**

3.1. MUL/DIV

Descripción de la instrucción

Multiplicación / División.

Ejemplos de uso de la instrucción

- **MUL dir1, dir2** ;Multiplica el contenido de la dirección 'dir1' por el contenido de la dirección 'dir2'.
- **DIV imd1, imd2** ;Divide el número 'imd1' entre el número 'imd2'.

3.2. NOP

Descripción de la instrucción

No hace nada.

Ejemplos de uso de la instrucción

- **NOP dir1, dir2** ;Pues eso, no hace nada. Pero los operandos se siguen evaluando, es decir, consume un ciclo de ejecución.

3.3. JMZ

Descripción de la instrucción

Salto condicional. Comprueba un número y salta si es cero.

Ejemplos de uso de la instrucción

- **JMZ dir1, dir2** ;Salta a la dirección 'dir1' si 'dir2' es cero.

3.4. JMN

Descripción de la instrucción

Salto condicional. Comprueba un número y salta si NO es cero.

Ejemplos de uso de la instrucción

- **JMN dir1, dir2** ;Salta a la dirección 'dir1' si 'dir2' NO es cero.

3.5. DJN

Descripción de la instrucción

Resta uno a un número y salta si el resultado de la resta NO es 0.

Ejemplos de uso de la instrucción

- **DJN dir1, dir2** ;Decrementa uno al contenido de 'dir1', si el resultado NO es cero, salta a la dirección 'dir2'.

3.6. SPL

Descripción de la instrucción

Crea un nuevo proceso en otra dirección.

Ejemplos de uso de la instrucción

- **SPL dir** ;Subdivisión del programa. Comienza un nuevo proceso en la dirección 'dir'.
- **SPL etiqueta** ;Subdivisión del programa, añadiéndose al proceso o procesos en ejecución el situado en la dirección de la 'etiqueta'.

3.7. CMP/SEQ

Descripción de la instrucción

Salto condicional. Compara dos números, si son iguales, se salta la siguiente instrucción.

Ejemplos de uso de la instrucción

- **SEQ dir1, dir2** ;Si el contenido de la dirección 'dir1', es igual al de 'dir2', no ejecuta la siguiente instrucción.

3.8. SNE

Descripción de la instrucción

Salto condicional. Compara dos números, si NO son iguales, se salta la siguiente instrucción.

Ejemplos de uso de la instrucción

- **SNE dir1, dir2** ;Si el contenido de la dirección 'dir1', NO es igual al de 'dir2', no ejecuta la siguiente instrucción.

4. Mi primer virus.

Lo primero es saber qué instrucciones están dentro de nuestro alcance y cuales sobrepasan nuestros conocimientos, es decir, hay que ser realistas, si es la primera vez que programas en un lenguaje ensamblador no es necesario que hagas un virus con decenas de saltos y líneas de código.
Y es tan sencillo como ver los ejemplos, entenderlos, coger uno que nos guste y mejorarlo.

Fernando

4.1. Ejemplo de ejecución instrucción a instrucción

Primera iteración del gusano

```
ADD #4, 3          ;Aquí comienza la ejecución.
MOV 2, @2
JMP -2
DAT #0, #0
```

Segunda iteración del gusano

```
ADD #4, 3
MOV 2, @2          ;Siguiente instrucción.
JMP -2
DAT #0, #4
```

Segunda iteración del gusano (detallada)

```
ADD #4, 3
MOV 2, @2 ;  ------,
JMP -2    ;       ||+2
DAT#0, #4;  <--'--,
...                ||
...                ||+4
...                ||
DAT#0, #4;  <----'
```

Tercera iteración del gusano

```
ADD #4, 3
MOV 2, @2
JMP -2          ;Siguiente instrucción.
DAT #0, #8
...
...
...
DAT #0, #4
...
...
...
DAT #0, #8
```

Cuarta iteración del gusano (salta de nuevo a la primera instrucción)

```
ADD #4, 3       ;Siguiente instrucción.
MOV 2, @2
JMP -2
DAT #0, #8
...
...
...
DAT #0, #4
...
...
...
DAT #0, #8
```

5. Estrategias.

Siempre se halla la eterna duda entre **FUERZA** y **VELOCIDAD**.
Rellenar rápidamente el núcleo o atacar a posiciones estratégicas.

En CoreWar existen **3 estrategias** fundamentales que, por analogía con el famoso juego, se les denominan **piedra, papel y tijera**.

Estrategias en CoreWar

1. **PAPEL:** hace múltiples copias de sí mismo lo más rápidamente posible, así sacrifica velocidad de ataque por resistencia.
   Esta estrategia vence a piedra pero pierde ante tijeras gracias a su gran capacidad de supervivencia aunque tienen una cierta tendencia al empate.

2. **PIEDRA:** bombardea direcciones de memoria a ciegas intentando matar rápidamente al mayor número de enemigos. Su reducido tamaño y sencillez los hace relativamente robustos y difíciles de localizar.
   Esta estrategia vence a tijeras pero pierde ante papel.

3. **TIJERA:** es la más avanzada. Comprueba posiciones de memoria a intervalos hasta localizar al guerrero rival. Una vez localizado generalmente sobrescriben su código con instrucciones que les obligan a generar nuevos procesos indefinidamente hasta quedar prácticamente bloqueados. Después proceden a eliminar todos los rivales.
   Esta estrategia generalmente vence al papel y pierde contra piedra, puesto que pierde tiempo atacando las posiciones de memoria alteradas por este último.

6. Ejemplos.

1. TRASGO:

```
MOV 0, 1 ;Copia el contenido de la dirección '0' en la dirección '1'.
         ;No puede ganar, sólo empatar.
```

2. ENANO BOMBARDERO (DRAW):

```
ADD #4, 3   ;Suma al contenido de la dirección '4', el entero '3'.
MOV 2, @2   ;Copia el contenido de la dirección '2' en la dirección contenida
            en la dirección '2'.
JMP -2      ;Salta a la dirección de memoria '-2'.
DAT #0, #0  ;Termina el proceso.
```

3. BLANCA-!NIEVES:

> **Nota**
>
> *El siguiente código es parte del virus 'BLANCA-NIEVES', observando atentamente, veremos que son una serie de 'ENANOS' encadenados de una forma sofisticada.*

```
Enano:     SPL Mudito
Sabiondo:  ADD #1328, 3
           MOV 2, @2
           JMP -2
           DAT #0, #0
Mudito:    ADD #516,3
           MOV 2, @2
           JMP -2
           DAT #0, #0
```

4. ZERG-RUSH:

```
salt  EQU #100      ;Etiqueta 'salt'. Establece una constante, cuyo valor será
                     100.
      MOV salt, 10  ;Copia el valor de la dirección de la etiqueta 'salt' en la
                     de direcciónmemoria '10'.
loop  MOV imp, @9   ;Etiqueta 'loop'.
      SPL @8        ;Crea un nuevo proceso en la dirección que contiene
                     la dirección de memoria '8'.
      ADD #100, 7   ;Suma '100' al contenido de la dirección '7'
      JMP loop      ;Salta a la etiqueta 'loop'
imp   MOV 0,1       ;Etiqueta 'imp'. Copia el contenido de la dirección '0' en
                     la dirección '1.
```

(Cortesía del Profesor José Luis Vázquez-Poletti).