CC41C Introducción al Hardware
Tarea 2 - Otoño 2006
Profesor: Luis Mateu

Una forma simple y portable de implementar lenguajes de programación es recurrir a un *intérprete de bytecode*. El compilador no produce código nativo para alguna máquina específica, si no que produce código para una máquina virtual de stack. Cada operación se representa mediante un byte, más opcionalmente algunos parámetros en bytes adicionales. Entonces se construye un intérprete de la máquina virtual en algún lenguaje eficiente como C. El típico intérprete tiene la siguiente forma:

```
for(;;) {
  switch(*pc++) {
    case OP: ... ejecución de OP ...
      break;
    ... etc. ...
  }
}
```

Un ejemplo de intérprete de bytecode es el que se muestra en el archivo `mcd-bytecode.c`. El archivo además contiene un arreglo con el bytecode que resuelve el problema del *máximo común divisor*. Las instrucciones que se definen en el intérprete son las que se necesitan para poder implementar el mcd y nada más. Un segundo ejemplo se encuentra en `sieve-bytecode.c`. Este archivo extiende el intérprete para poder calcular los números primos.

La principal desventaja de los intérpretes de bytecode es que son lentos. La principal ventaja es que son portables de una máquina a otra siempre y cuando posea un compilador de C.

Un *intérprete de threaded-code* es un intérprete más eficiente que uno de bytecode. El compilador traduce los programas a una secuencia de palabras de 32 bits, en vez de bytes. En vez de utilizar un código para representar una operación se utiliza directamente la dirección del código que ejecuta la operación. La desventaja es que ahora el intérprete debe ser escrito en assembler, puesto que no es posible hacerlo en C estándar (auque GCC si posee extensiones que permiten implementar este tipo de intérpretes). Por lo tanto un intérprete de threaded-code no es portable, aunque es mucho más fácil de implementar que un compilador que genera código nativo. Es así como el intérprete de threaded-code para una x86 tiene la siguiente forma:

```
# suponiendo que sp es %ecx
OP:     addl $4, %ebx      # pc++
        movl (%ebx), %eax  # dir. de salto
        ... ejecución de OP ...
        jmp *%eax  # salto indirecto
... lo mismo para el resto de las operaciones ...
```

Un ejemplo de intérprete de threaded-code es `mcd-threaded.s`. Este archivo contiene el intérprete y el threaded-code para calcular el mcd. El programa test-mcd-threaded.c permite ejecutar este intérprete.

Recupere el archivo `t2.tgz` de la página Web del curso, descomprímalo y estudie cuidadosamente los intérpretes mencionados: `mcd-bytecode.c`, `sieve-bytecode.c` y la dupla `mcd-threaded.s` + `test-mcd-threaded.c`. Compile y ejecute estos programas.

Parte a.-

Modifique el archivo `mcd-threaded.c` para que se despliegue después de ejecutar cada instrucción el valor del contador de programa, el puntero a la pila y 3 valores del tope de la pila. Ud. debe implementar el despliegue *llamando a printf*. Use un formato similar al de `mcd-bytecode.c`. No olvide resguardar los registros que pueden ser modificados por *printf*.

Parte b.-

Extienda el intérprete de threaded-code con las instrucciones que faltan para calcular los números primos. Base su implementación en el archivo `sieve-threaded.s.tmpl`. No haga cambios en la parte que ya se encuentra implementada. Pruebe su programa utilizando test-sieve-threaded.c.

Indicación: No vacile en compilar con las opciones -S y -O para ver qué código genera GCC para compilar los programas en C (-S genera el assembler y -O optimiza el código). El objetivo de esta tarea no es adiestrarlo en la programación en assembler x86, si no que Ud. sea capaz de entender los programas en assembler x86.

Entrega

El plazo de entrega vence el Miércoles 31 de Mayo. Se descontará medio punto por día de atraso. Entregue en U-cursos un archivo *tar comprimido con gzip* con todos los archivos que modificó.