Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

Minitarea 4

Autor: Lukas Pavez
RUT: 19.401.577-1
Profesor: Pablo Guerrero P.
Auxiliar: Matías Torrealba A.
Ayudantes: Gabriel Chandía G.
Gaspar Ricci
Fecha de entrega: 12/11/2018

1. P1

1.1. Int

Se puede ver en el código que se guarda el valor inmediato del entero (en este caso $5) en el registro de la pila %rbp y luego mueve eso a %eax, por lo que los int se representan como un valor inmediato.

1.2. Float

Se puede ver que para los float assembler crea una subrutina .LC0, y mueve el valor guardado en el registro %rip al %xmm0, luego del %xmm0 al %rbp, del %rbp al %xmm0 y finalmente del %xmm0 al %eax. Por lo que se pudo determinar que assembler trata a los float como un .long y utiliza otro registro para manejarlos (los %xmm).

1.3. Char

Como con los int, con los char se guarda el valor inmediato del caracter (según la tabla ascii) en el registro de la pila %rbp, con la diferencia de que con los char se mueve un byte al registro y con los int se mueve un long.

1.4. Punteros

Como lo que se hizo en el programa fue crear una variable int y luego un puntero a esa variable, se puede ver que, luego de guardar el valor del entero en $-20(\%rbp)$, utiliza la operacion leaq (load effective address) entre los registros $-20(\%rbp)$ y %rax, por lo que se guarda la direccion de memoria de $-20(\%rbp)$ en el registro %rax, y luego guarda el registro %rax en %rbp, por lo que pudo determinar que registros utiliza assembler para los punteros.

1.5. Arreglos de char

Se creo el arreglo de char "hello", y en el .s se puede ver que se realizan operaciones entre los registros %fs, %rax y %rbp, y luego mueve un numero grande (en este caso $1819043176) al registro de pila %rbp, y despues mueve otro numero ($111) a %rbp, por lo que los arreglos de char se pueden ver como un numero muy grande (ademas de las operaciones necesarias con los registros).

2. P2

2.1. If

En assembler, se hace una comparacion entre un entero y el valor del registro %rbp donde esta una variable creada al principio, y dependiendo del resultado, puede saltar a .L2 que mueve un numero a %eax (o el valor de %rbp a %eax).

2.2. Switch-case

En esta operación se crea un .LN para cada caso y se tiene que buscar cual es el que se tiene.

2.3. While

La implementacion de esta operación tiene 3 partes principales: la que comienza inicializando las variables (.LFB0), otra que ejecuta lo que se encuentra dentro del while (.L3), y la última que ejecuta lo que pasa después del while (.L2), lo que ocurre es que desde .LBF0 se salta a .L2, y al principio de .L2 se hace la comparación de la condición del while, si no se cumple salta a .L3, y continua con .L2, hasta que termina.

2.4. For

Se puede ver que se hace lo mismo que la operación while, con la diferencia de que en este caso se utilizó una variable más, por lo que se agregaron 2 líneas de código (una para iniciar la variable en 0 y otra para sumarle 1 dentro del for).

2.5. Eficiencia

Se pudo concluir que en cuanto a la eficiencia, entre if y switch-case, que es mejor utilizar if, ya que esta operación hace la comparación y luego salta a lo que se tiene que ejecutar, mientras que switch-case tiene que buscar que caso se va a ejecutar. Y entre for y while, la implementación es la misma, por lo que preferir uno no haría que el programa sea más eficiente.