Taller de Programación de Sistemas CC207

NRC 07073

MARIA ELENA ROMERO GASTELU

2013-B Sección D02

Práctica #6

Alan Andrés Sánchez Castro 208697345

Modificación de clases:

Se modificó el archivo **Util.py** añadiendo varias funciones:

- ***comp2(n,b)*** *Retorna el valor del complemento a 2 de n a b bits*
    - n es el número al que se le quiere sacar su complemento a 2 y b es la cantidad de bits que se querrán obtener.
    - Funciona haciendo una serie de operaciones a nivel de bit *((n^((1<<b)-1))+1)*. Lo que hace es invertir todos los bits con la operación xor, tomando n a b bits, por ejemplo, 2 a 4 bits se tomaría como 0b0010 y no solamente 0b10, y después le suma 1.
- Los diccionarios que se usarán para calcular los respectivos códigos xb
    - _XBRR={"X":0,"Y":1,"SP":2,"PC":3}
    - _XBAA={"A":0,"B":1,"D":2}

Además se añadieron las funciones necesarias para obtener el código máquina generado por los siguientes modos de direccionamiento:

- **Indizado de 5 bits**
- Sólo se necesita calcular el byte xb, lo cual se hace de siguiente manera:
  * rr0nnnnn
  -Se inicializa xb en 0
  -Se le añaden el par de bits rr 6 posiciones a la izquierda (xb|=_xbrr[r]<<6)
  -Si nnnnn es positivo, se le añade a xb, sino, se añade su complemento a 2:
    - if n>=0:
    - xb|=n
    - else:
    - xb|=comp2(-n,5)
- **Indizado de 9 bits**
- Se necesitan calcular el byte xb y la representación hexadecimal del valor del operando.
- Para el byte xb, se utiliza el siguiente algoritmo:
  * 111rr0zs
  La z siempre será 0 y s será 1 si el signo del operando es negativo, 0 en caso contrario.
  -Se inicializa xb en 0
  -Se le añaden las constantes (xb|=0xe0, lo que es lo mismo a xb|=0b11100000)
  -Se le añaden el par de bits rr 3 posiciones a la izquierda (xb|=_xbrr[r]<<3)
  -Si el valor del operando es negativo, se le añade el bit de signo a xb
    - if n<0:
    - xb|=1
- Para el valor del operando, si n es positivo simplemente se obtiene su representación hexadecimal, en caso contrario primero se obtiene su complemento a 2 con 8 bits y después la representación hexadecimal de eso.
- **Indizado de 16 bits**
- Se necesita calcular el byte xb y 2 bytes de la representación hexadecimal del valor del operando
- Para el byte xb, se utiliza el siguiente algoritmo:
  *111rr0zs

La z siempre será 1 y s siempre será 0
-Se inicializa xb en 0
-Se añaden las constantes (xb|=0xe2, lo que es lo mismo a xb|=0b11100010)
-Se le añaden el par de bits rr 3 posiciones a la izquierda (xb|=_xbrr[r]<<3)

- **Indizado Pre/Post Incremento/Decremento**
- Sólo se necesita calcular el byte xb, lo cual se hace de la siguiente manera:
  *rr1pnnnn
  -Se inicializa xb en 0
  -Se le añade la constante (xb|=0x20, lo que es lo mismo a xb|=0b00100000)
  - Se le añaden el par de bits rr 6 posiciones a la izquierda (xb|=_xbrr[r]<<6)
  - Si es post, se añade su bit correspondiente (xb|=1<<4)
  -Si el signo es positivo (incremento) entonces se añade n-1 (xb|=(n-1))
  -En caso contrario se añade 0x10-n (xb|=0x10-n)
- **Indizado de Acumulador**
- Sólo se necesita calcular el byte xb, para lo que se utiliza el siguiente algoritmo:
  *111rr1aa
  -Se inicializa xb en 0
  -Se le añaden las constantes (xb|=0xe4, lo que es lo mismo a xb|=0b11100100)
  -Se le añaden el par de bits rr 3 posiciones a la izquierda (xb|=_xbrr[r]<<3)
  -Se le añaden el par de bits aa (xb|=_xbaa[a])

Para todos estos modos, el código final generado es la concatenación del código de máquina de la instrucción más el código xb (su representación hexadecimal), además de la representación hexadecimal, en caso de ser necesario, del valor numérico del operando.