102685 Arquitectura de Computadors i Perifèrics
PRÀCTICA Nº 2 CURS 2020/2021

ESPAI DE DADES, ESPAI DE CODI I ADREÇAMENTS

Objectius de la pràctica

En aquesta sessió de pràctiques treballarem dos temes:

1) Com es declaren les variables en el codi assemblador del microprocessador Cortex M0.

2) Com accedir o modificar el contingut d'aquestes variables, utilitzant els diferents modes de adreçament del microprocessador (explicats a classe i en el document del repertori d'instruccions de ARM situat al CV).

Generació de codi binari

Els programes fonts, escrits en llenguatge assemblador o en un d'alt nivell (com C/C++ o Java), es compilen per generar el seu codi executable (codi binari que por ser executat pel microprocessador). Aquest codi es divideix en dos seccions importants: La de **dades**, que conté les variables (i els seus valors) i la de **codi**, que conté les instruccions del codi màquina que executa el processador.
Quan escrivim un programa utilitzant el llenguatge **assemblador de GNU**, que es el que s'utilitza per a la família de processadors ARM, el començament de la secció de codi s'identifica amb la directiva **.text** i el de la de dades, amb la directiva **.data**.
La secció de codi ja la vam treballar en la primera sessió de pràctiques, van afegir instruccions assemblador després de l'etiqueta **main** d'aquesta secció. La secció de dades s'ha de situar al principi del nostre codi assemblador i la seva sintaxis és:

```
        .data
<nom var>:
        .<tipus variable> <valor o valors de la variable>
```

A on (compte en els punts davant dels identificadors),

.data: Com ja s'ha vist, directiva que defineix el començament de la secció de dades.
<nom var>: Es el nom de la variable.
.<tipus variable>: És el tipus de la variable. Alguns d'aquests tipus són: **.int** per sencers, **.ascii** per cadenes de caràcters, **.byte** per caràcters (o bytes), **.float** i **.double** per nombres en punt flotant (ARM segueix el format IEEE per aquests tipus de dades).
<valor o valors de la variable>: És el valor o valors que li assignem a la variable. En cas que definim una variable tipus array (o vector), els valors de cadascun dels seus índexs (numerats del 0 al N-1, com C/C++) estaran separats per comes, per tant la declaració de la part del tipus d'una variable array serà:

.<tipus variable> valor_index_0, valor_index_1,...,valor_index_N-1

Per exemple, si volem declarar una variable array de 3 sencers (anomenada **var_int**), situarem al principi del nostre codi assemblador la següent declaració:

```
  .data
var_int:
  .int 1,3,7
```

Un altra qüestió important, a l'hora de treballar amb els modes de adreçament, és com obtenir l'adreça de memòria d'una variable. Per realitzar aquesta tasca podem utilitzar la següent instrucció: **ldr RX,=nom_var**, a on **RX** pot ser qualsevol registre de propòsit general del processador Cortex M0 i **nom_var** és el nom de la variable. Per exemple, per guardar l'adreça de la variable **var_int** (declarada abans) en el registre R3, utilitzaríem la següent instrucció:
ldr R3,=var_int.

Representació de les bases dels valors dels registres

A la Figura 1 es mostra com podem escollir les diferents bases dels valors que pot prendre un registre. Ho podrem escollir a dins de la finestra de control.

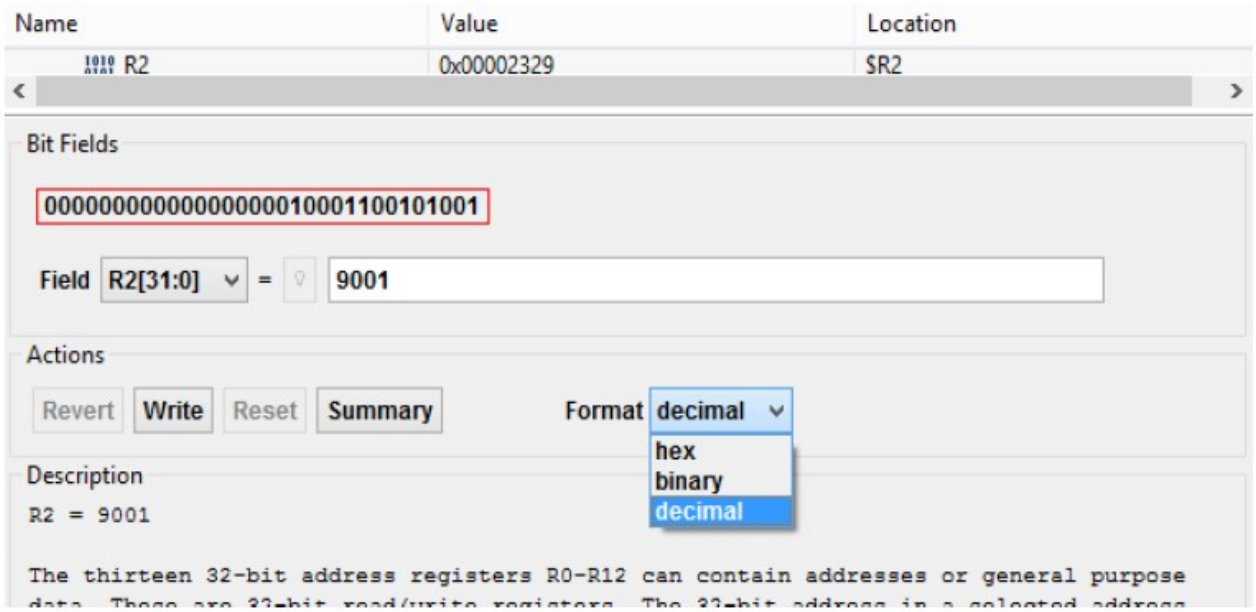

Figura 1: Bases de representació dels registres

Exercici

En el CV s'ha situat l'esquelet d'un programa en codi assemblador del processador Cortex M0, que s'ha de completar (als requadres). Aquest programa es una implementació de l'algoritme d'ordenació de **la bombolla.** Aquest algoritme va comparant cada element d'una llista **N** a ordenar amb el següent (element **N[j]** amb el **N[j+1]**). Si aquests dos elements no estan ordenats, llavors s'intercanvien les seves posicions. Per exemple, si volem ordenar la llista **N** de menor a major i **N[j] > N[j+1]**, llavors l'element de la posició **j** passa a la **j+1** i aquest últim a la posició **j**. Aquest procediment es

repeteix fins que tots els elements de la llista estan ordenats.

Documentació

- Al Campus Virtual: Documents del codi assemblador del Cortex M0
- Directives assemblador GNU: https://www.sourceware.org/binutils/docs-2.12/as.info/Pseudo-Ops.html#Pseudo%20Ops
- Algorisme ordenació de la bombolla: https://es.wikipedia.org/wiki/Ordenamiento_de_burbuja