Andiamo avanti con la programmazione non sotto interruzione. L’esempio da presentare oggi è una funzione di libreria che fa la conversione da stringa a intero, Str2int. L’idea è avere a disposizione un buffer che contiene la sequenza dei caratteri letti dal terminale e dai quali estrarre la rappresentazione del numero. L’indirizzo del buffer deve trovarsi all’interno del registro R0 (e in C equivarrebbe a un puntatore a char) e il risultato viene messo nel registro 1.

STM SP, R2, LP //si effettua innanzitutto il salvataggio dei registri (esclusi R0 e R1 perché servono per i parametri)

MOV R1, #0

Repeat: LDRB R2, [R0], #+1 //si utilizza l’indirizzo in R0 per effettuare la lettura e poi si fa l’auto-incremento.

//vogliamo verificare di avere a che fare con una cifra decimale [0-9], per questo effettuiamo una sottrazione

SUBS R2, R2, #48 //la S dopo SUB è lì perché vengono modificati i bit di stato,

//ci interessa che il valore sia >= 0

LDMLT SP, R2, PC //per default un qualsiasi carattere che non sia una cifra fa ritornare il valore //corrente, che all’inizio è 0

CMP R2, #9 //controlliamo che R2 non sia > 9

LDMGT SP, r2, PC

ADD R1, R1, R1 lsl #2 //moltiplicazione per 5 di R1

ADD R1, R2, R1 lsl #1 //moltiplicazione per 2 di R1 e ci si somma R2

B repeat

Funzione inversa int2str(int val, char \* buffer, int dim) //val è in R0, il puntatore al buffer in R1 e le sue dimensioni in R2.

STM SP, R3, R4, R5, LP

SUBS R2, R2, #1

LDMLE R3, R4, R5, PC

MOV R3, #0 //inizializzo a 0 R3

STRB R3, [R1, +R2] //Porto l’ultimo elemento del buffer = R3 (cioè = 0, che fa da terminatore)

MOV R3, #1 //numero di cifre da stampare

MOV R4, R2 //muoviamo i registri 1 e 2 in 4 e 5 per poter chiamare la divide

MOV R5, R1

MOV R1, #10 //mettiamo 10 come divisore perché dobbiamo prendere cifra per cifra

while:

SUBS R4, R4, #1

LDMLT SP, R3, R4, R5, PC //se raggiungiamo la fine del buffer dobbiamo fermarci (e vengono stampate solo le cifre meno significative)

BL divide //chiamo la divisione per 10

ADD R0, #48 //dopo la divide, in R0 è presente il resto quindi per renderlo ASCII ci basta aggiungere +48

STRB R0, [R5, +R4] //si aggiunge la lettera alla fine del buffer ancora libero (che sarà poi scandito dall’inizio alla fine)

CMP R2, #0 //confrontiamo il quoziente con 0, se è = 0 abbiamo finito il nostro numero

LDMLE SP, R3, R4, R5, PC

MOV R0, R2 // si sposta in quoziente in R0, così può essere nuovamente diviso

B while