Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

HTA-FR – Kurs Telekommunikation

Verfasser:
D. Gachet / HTA-FR - Telekommunikation

Embedded systems 1 und 2

Praktische Arbeiten PA2
Einführung in den ARM-Assembler

Klasse T-2 // 2016-2017

Zielsetzungen

Zielsetzungen:

- **Am Ende der Laborübung sind die Studierenden in der Lage:**
  - Die wichtigsten Adressierungsmodi des µP ARM zu beschreiben
  - Einen grundlegenden Algorithmus in Assembler zu entwickeln (planen, codieren und testen), welche die Primzahlen bis 100 berechnet und darstellt
  - Den Prozess der Assemblierung, des Linkens und des Debuggens zu beherrschen

Dauer :

- 1 Laborübung (4 Stunden)

Bericht und Bewertung

- Laborbericht mit Quellcode

Zu realisierende Arbeit

- **Entwickeln Sie einen Assemblercode, welche die Primzahlen bis 100 gemäss das Sieb des Eratosthenes berechnen und darstellen kann.**

```
#define MAX     100

bool is_a_prime_number[MAX];

void prime_number_generator()
{
        // 1st mark all numbers as prime number
        for (int i=0; i<MAX; i++) {
                is_a_prime_number[i] = true;
        }

        // 2nd mark all multiples as not a prime number
        for (int i=2; i<MAX; i++) {
                for (int j=i*2; j<MAX; j+=i) {
                        is_a_prime_number[j] = false;
                }
        }

        // 3rd print all prime numbers
        for (int i=2; i<MAX; i++) {
                if (is_a_prime_number[i]) {
                        printf ("%d\n", i);
                }
        }
}
```

Updates und Bediengungen

- **Folgende Ausführungsbedingungen sind gegeben**
  - Das Skelett des Projeks befindet sich in der zentralen Ablage.
  - Um es herunter zu laden, geben Sie den folgenden Git-Befehl ein:

    ```
    $ cd ~/workspace/se12/tp
    $ git pull upstream master
    ```

  - Der Algorithmus soll in die Datei "*main.S*" integriert werden

- **Der Code und der Bericht müssen in die zentrale Ablage hochgeladen werden.**
  - *Code: .../tp/tp.02*
  - *Bericht: ../tp/tp.02/doc/report.pdf*

- **Termin**
  - Der Bericht und der Code sind am gleichen Tag um 24:00 Uhr abzugeben.

Fragen

Beantworten Sie während der Realisierung der praktischen Arbeit die folgenden Fragen:

1. Nennen Sie die gebräuchlichsten Adressierungsmodi, die durch den µP ARM unterstützt werden.
2. Wie wird der von Ihnen geschriebene Assemblercode binär codiert?
3. Welche Werkzeug ermöglicht die Applikation von generieten Code zu erzeugen?
4. Kann man den Algorithmus schneller machen? Falls ja, wie?
5. Kann man den Algorithmus ändern um wenig Speicher zu brauchen? Falls ja, wie?

Codierungshinweise

1. Die Adresse einer Variablen kann einfach mit dem folgenden Assemblerbefehl in ein Register geladen werden:

   ```
   LDR  Rn, =variable
   ```

2. Eine Konstante grösser 255 kann mit dem folgenden Assemblerbefehl in ein Register geladen werden:

   ```
   LDR  Rn, =<constante>
   ```

3. Die folgenden Assemblerbefehle erlauben das Datentransfer zwischen Registers und Hauptspeicher

   ```
   LDR<sz> Rx, [Ry]
   STR<sz> Rx, [Ry
   ```

   `<sz>` Datengrösse

   - `B` 8-Bitworte
   - `H` 16-Bitworte
   - `blank` 32-Bitworte

Codierungshinweise (II)

- Die Variablen lassen sich mithilfe der Kombination der folgenden Befehle testen:

```
CMP       Rn, Rm
B<cc>     <etiquette>
```

- Wenn der zu testende Wert kleiner als 256 ist, lässt sich der folgende Testbefehl verwenden:

```
CMP       Rn, #<val>
```

Mnemonic	Meaning
HI	Unsigned higher
HS	Unsigned higher or same
EQ	Equal
NE	Not equal
LS	Unsigned lower or same
LO	Unsigned lower

Codierungshinweise (III)

- Die Codierung der hexadezimalen Werte erfolgt mithilfe eines `0x<val>`

- Die Anzeige eines Strings auf dem Terminal lässt sich mit dem folgendem Befehlssatz erreichen

```
LDR     R0, =<format>    // printf formatting string
LDR     R1, =<1st-arg>   // printf 1st argument / optional
LDR     R2, =<2nd-arg>   // printf 2nd argument / optional
LDR     R3, =<3rd-arg>   // printf 3d argument / optional
BL      printf
```

- Achtung: nach dem Aufruf der Funktion `printf`, die Registers R0 bis R3 sind sehr wahrscheinlich geändert