TP n°2

Informatique embarquée

Vincent Ruello

À rendre avant mercredi 25/10/2023 - 16:00 (CEST)

Le but de ce TP est de faire clignoter la LED intégrée d'une carte Arduino UNO R3 en assembleur AVR.

Contexte

La carte Arduino UNO est une plateforme open-source de développement dont un schéma électrique simplifié est disponible ici[1]. Elle est composée de deux microcontrôleurs AVR : un ATmega16U2 (connecté au port USB) et un ATmega328P (connecté à la plupart des broches externes). La documentation de ce dernier est disponible ici[2]. Son jeu d'instructions est détaillé ici[3].

Partie 1. Premiers pas avec l'ATmega328P

Cette partie a pour but d'écrire un premier programme qui allume la LED intégrée de la carte Arduino et de le télécharger sur le microcontrôleur.

1. D'après le schéma électrique de la carte Arduino, sur quelle broche du microcontrôleur est branchée la LED intégrée de la carte Arduino ?

   *Une broche est identifiée par **une lettre** et **un numéro**. Exemple : D4.*

2. D'après la *datasheet* du microcontrôleur, quelles sont les étapes nécessaires afin d'appliquer un potentiel *high* à une GPIO en sortie ?

3. Écrire un programme en assembleur AVR qui allume la LED intégrée de la carte puis attend indéfiniement.

4. Assembler ce programme et obtenir un fichier objet à l'aide de `avr-as` :
   ```
   avr-as main.s -o main.o
   ```

5. Faire la résolution des liens du programme et obtenir un fichier exécutable au format ELF à l'aide de `avr-ld` :
   ```
   avr-ld main.o -o program.elf
   ```

1 https://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf
2 http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf
3 https://ww1.microchip.com/downloads/en/DeviceDoc/AVR-Instruction-Set-Manual-DS40002198A.pdf

6. Vérifier les instructions présentes dans le binaire avec `avr-objdump` (option `-d`).

7. Convertir votre exécutable au format ihex (Intel hex format) avec `avr-objcopy`.
```
avr-objcopy -O ihex program.elf program.hex
```

8. Flasher le fichier au format ihex sur la mémoire flash du microcontrôleur avec `avrdude`. Le programmeur à utiliser est `arduino`. Le *baud rate* utilisé est 115200.
```
avrdude -patmega328p -carduino -P/dev/ttyACM0 -b115200
      -Uflash:w:program.hex
```

9. Décrire dans un fichier `Makefile` les étapes précédentes (4, 5, 7, 8).

Partie 2. Boucle d'attente

Sur la carte Arduino, l'ATmega328P est cadencé à 16MHz. A cette fréquence, en utilisant le jeu d'instruction AVRe, on souhaite écrire une fonction `sleep_500_ms` qui retourne une fois que 500ms se sont écoulées en utilisant une boucle d'attente.

L'idée générale est de convertir le temps qu'on souhaite attendre en nombre de cycles CPU, puis d'exécuter le bon nombre d'instructions pour qu'il se produise le nombre de cycles souhaité entre l'appel et le retour de la fonction. Dans la plupart des cas, on implémente ce type de mécanisme avec une boucle qui décrémente un compteur initialisé à une valeur choisie en fonction du temps qu'on souhaite attendre.

Registres utilisables sans précautions particulière : `r18-r28`, `r30-r31`.

1. Écrire en assembleur AVR un programme qui initialise le registre r24 à 0xff (255) puis décrémente sa valeur jusqu'à atteindre 0x00 (0).

2. D'après la documentation, combien de cycles CPU sont utilisés dans ce programme ? En déduire une formule qui exprime la valeur initiale du compteur en fonction du nombre de cycles CPU.

3. Calculer le temps écoulé entre le début et la « fin » du programme. Comment faire pour augmenter ce temps ?

4. Réaliser en assembleur AVR un programme qui décrémente une valeur codée sur 3 octets stockée dans r18, r19 et r20 (r20 est l'octet de poids fort, r18 est l'octet de poids faible). La valeur doit être initialisée à 0xffffff et la valeur finale est 0x000000. Il est possible de réaliser ce programme en 7 instructions, en utilisant notamment `subi` et `sbci`.

5. Combien de cycles CPU sont utilisés dans ce programme ? En déduire une formule qui exprime la valeur initiale du compteur en fonction du nombre de cycles CPU.

6. Combien de cycles CPU se produisent en 500 ms ? En utilisant un compteur sur 3 octets tel qu'implémenté à la question 5, que doit être la valeur initiale du compteur pour que 500ms se déroulent entre le début et la fin du programme ?

7. Écrire la fonction `sleep_500_ms`. Pour rappel, une fois le registre `SP` (Stack Pointer) initialisé, il est possible d'utiliser l'instruction `rcall <label>` pour appeler une fonction. L'instruction `ret` permet, depuis le code de la fonction, de revenir à l'instruction suivant le `call`.

8. Écrire un programme qui fait clignoter la LED intégrée toutes les 500ms à l'aide de la fonction `sleep_500_ms`. On veillera à initialiser le registre 16 bits SP (SPL et SPH) à l'adresse la plus élevée en mémoire données. Télécharger le programme sur le microcontrôleur.