Enginyeria de Sistemes TIC

Dispositius Programables

Pràctica 3 - Generador de polsos

Francisco del Águila López

Octubre 2011
Escola Politècnica Superior d'Enginyeria de Manresa
Universitat politècnica de Catalunya

1 Objectiu

L'objectiu d'aquesta tercera sessió és aprofundir en la utilització de l'assemblador i els ports d'entrada / sortida del microcontrolador.

2 Introducció teòrica

2.1 Crida a subrutines

En general, tots els microcontroladors tenen instruccions de crida a subrutina. En el cas dels AVR són **call**, **rcall** i **icall** amb la seva corresponen instrucció de retorn **ret**. Una crida a una subrutina es comporta similar a la instrucció de salt incondicional **jmp**, **rjmp** però amb la peculiaritat que una vegada s'ha executat un tros de codi concret l'execució del programa continua a la posició següent d'on es troba la instrucció **call**, **rcall** o **icall**. Això és útil per executar un tros de codi concret que es pot anar repetint en diferents llocs del programa principal. Una descripció més detallada d'aquestes instruccions les trobem a [Instr].

El funcionament d'aquestes crides, implica que el microcontrolador ha de recordar en quina posició de memòria de programa (Program Counter: **PC**) s'ha produït aquesta crida. De tal manera que un cop s'executa el codi de la subrutina i es troba la instrucció **ret** al final, es recupera la posició que tenia el **PC**. El lloc on es guarda aquesta posició de memòria és l'*stack*. Una descripció més detallada de l'*stack* la trobem a l'apartat 7.5 pàg 13 de [ATmega328p]. Com s'indica a la documentació, l'*stack* també es pot fer servir per guardar dades amb les instruccions **push** i **pop**.

2.2 Macros

Les macros serveixen per definir un tros de codi que l'assemblador anirà enganxant allà on es faci referència a la macro. Per tant és una tasca que realitza l'assemblador en el moment de crear el codi executable. Simplement, estalvia la feina al programador d'anar repetint el tros de codi en el moment de crear el codi font. Una descripció la trobem a la directiva .macro de [AS].

Quan a dins d'una macro s'han de fer servir etiquetes, *labels*, a adreces de programa s'ha d'anar amb compte ja que aquestes labels no es poden repetir. Una manera de solucionar aquest problema és fer servir etiquetes locals. Aquestes etiquetes es defineixen amb un nombre seguit de dos punts (xx:) i quan es fa referència a aquestes labels es pot afegir la lletra *b* o *f* per indicar al assemblador que és la primera etiqueta *backwards* o *fordward* que trobi en el codi. Un exemple de funcionament es veu en el codi *segon.S*.

2.3 Ports d'entrada / sortida

Els AVR disposen de diferents ports d'entrada / sortida que estan mapejats a la memòria d'entrada / sortida, que a la vegada està solapada a la memòria de dades. Aquests ports estan connectats a les potes del circuit integrat. La configuració d'aquests ports es troba a l'apartat 14 pàg. 77 de [ATmega328p].

3 Exemples de programa

Els següents exemples de programa s'encarreguen de encendre i apagar el led de l'Arduino per tal que es pugui veure. Els dos fan el mateix amb estructures diferents.

El primer.S fa servir la crida a subrutina:

```
.set DDRB_o , 0x4
.equ PORTB_o , 0x5

.global main

waitabit:
        ldi r19,41
wait3:  ldi r18,0xFF
wait2:  ldi r17,0xFF
wait1:  subi r17,0x01
        brne wait1
        subi r18,0x01
        brne wait2
        subi r19,0x01
        brne wait3
```

```
        ret

main:   ldi r16,0xFF
        out DDRB_o,r16
loop:   ldi r16,0x00
        out PORTB_o,r16
        rcall   waitabit
        ldi r16,0xFF
        out     PORTB_o,r16
        rcall   waitabit
        rjmp    loop        /* loop forever */
```

Segon.S fa servir macros:

```
.set DDRB_o , 0x4
.equ PORTB_o , 0x5

.macro waitabit tot
        ldi r19,\tot
3:      ldi r18,0xFF
2:      ldi r17,0xFF
1:      subi r17,0x01
        brne 1b
        subi r18,0x01
        brne 2b
        subi r19,0x01
        brne 3b
.endm

.global main

main:   ldi r16,0xFF
        out DDRB_o,r16
loop:   ldi r16,0x00
        out PORTB_o,r16
        waitabit 41
        ldi     r16,0xFF
        out     PORTB_o,r16
        waitabit 41
        rjmp    loop        /* loop forever */
```

El programa amb macros fa servir etiquetes locals. La macro està definida amb un paràmetre (*tot*) al que se li dóna el valor cada vegada que en el programa principal es crida a la macro.

Una manera de veure les subrutines i les macros és com si definíssim una nova instrucció basada en les instruccions bàsiques existents.

4 Estudi previ

1. El tros de codi format per la macro i per la subrutina són iguals. Deixant com a paràmetre el valor que es carrega al registre *r19* en la primera instrucció, calcula el nombre de cicles màquina que es tarda en executar aquest tros de codi.
2. Tenint en compte que el rellotge del microcontrolador és de 16 MHz, calcula el valor que s'ha de carregar al registre *r19* per tal que el temps total que tarda en executar-se aquest tros de codi sigui de 0.5 s.
3. Una manera per generar un to audible amb una sortida digital és generar una ona quadrada de freqüència audible (per exemple 1KHz). Això ho podem aconseguir commutant entre 0 i 1 una de les potes de sortida de l'Arduino. Comprova si la macro / subrutina *waitabit* serviria per poder generar una ona quadrada de 1KHz en una de les potes de l'Arduino. Si consideres oportú, fes les modificacions a *waitabit* generant *waitato* per tal que fer servir la macro / subritina per generar el to.
4. Defineix les configuracions (en codi assemblador) per configurar l'Arduino amb la pota 8 de les sortides digitals com sortida, la pota 13 (led) com sortida i la pota 7 com entrada. Aprofiteu els pull-ups interns.
5. Genera un programa en assemblador que generi un to de 1 KHz per la pota 8 de l'Arduino. Anomena'l prac3_1.S.
6. Així com en la macro que s'ha definit a segon.S podem passar un paràmetre cada vegada que la fem servir en el programa principal. Proposa una manera de poder passar un paràmetre quan es fa servir la crida a la subrutina *waitabit* en el programa principal.
7. Genera un programa en assemblador que generi un to de 1 KHz per la pota 8 de l'Arduino i al mateix temps encengui i apagui el led a una freqüència de 1s. Anomena'l prac3_2.S.
8. Genera un programa que quan comprovi que a la pota 7 de l'Arduino hi ha una tensió de 0V encengui el led durant 2s i el torni a apagar. Si passats els 2s es mantenen els 0V ha de tornar a encendre 2s més. Anomena'l prac3_3.S.
9. Modifica el programa anterior amb l'objectiu que es generi un to de 1KHz durant 2s a la pota 8 de l'Arduino quan hi ha una tensió de 0V a la pota 7. Anomena'l prac3_4.S.

5 Treball pràctic

El treball al laboratori consisteix en la comprovació pràctica de les tasques de l'estudi previ.

5.1 Assemblat del codi font

Per assemblar el nostre codi font s'ha de cridar a la comanda

```
avr-gcc -mmcu=atmega328p -o prova.elf prova.S
```

on el paràmetre **-mmcu** indica el model de microcontrolador que estem fent servir, **-o** indica el fitxer final generat (format *elf*) i l'últim paràmetre és directament el fitxer amb el codi font.

Un paràmetre que pot ser d'utilitat és **-E**, en aquest cas es realitza l'assemblat estrictament. Pot ser d'utilitat per analitzar possibles errors.

```
avr-gcc -mmcu=atmega328p -o prova.asm -E prova.S
```

Per convertir el format *elf* a *ihex*, la comanda és

```
avr-objcopy --output-target=ihex prova.elf prova.hex
```

Una possibilitat molt important és el procés de des-assemblat. En aquest cas, a partir d'un fitxer executable de tipus *elf* s'obté un fitxer en codi font. Òbviament la informació extra que conté el codi font original ha desaparegut. La comanda és

```
avr-objdump -S prova.elf > prova.disam
```

5.2 Transferència dels fitxers executables (Avrdude)

Quan es connecta l'Arduino a l'ordinador al port USB, s'ha de comprovar que s'ha detectat el port de comunicació i per tant està reconegut pel sistema. Això es pot comprovar amb l'ordre *dmesg*. A les línies finals hauria d'aparèixer un nou dispositiu amb identificació **ttyACM0.**

Per comprovar que l'Arduino està operatiu i disposat a acceptar ordres, la comanda és

```
avrdude -c arduino -P /dev/ttyACM0 -p m328p
```

on els paràmetres indiquen que el tipus de programador que fa servir l'*avrdude* és el que inclou directament el kit Arduino, que el port de comunicacions és el nou port USB detectat i que el dispositiu amb el que el treballa és un ATmega328p. Aquests paràmetres es poden consultar en el manual *man avrdude*.

Per fer les transferències dels fitxers binaris es fa servir l'opció **-U**(consulteu el manual). Els paràmetres d'aquesta opció són la memòria a la que s'accedeix, si es fa lectura o

escriptura, el fitxer que es vol transferir, el format del fitxer. Un exemple per llegit la memòria de programa i crear un fitxer Intel Hex amb el seu contingut seria

```
avrdude -c arduino -P /dev/ttyACM0 -p m328p -U flash:r:prova.hex:i
```

5.3 Tasques

1. Assembla *primer.S* i *segon.S*, transfereix-los a l'Arduino i comprova que fan el mateix. Des-assembla els dos programes i compara'ls. Descriu les diferències.
2. Comprova el correcte funcionament de *prac3_1.S*.
3. Comprova el correcte funcionament de *prac3_2.S*.
4. Comprova el correcte funcionament de *prac3_3.S*.
5. Comprova el correcte funcionament de *prac3_4.S*.

Referències

[Ard] Arduino UNO - `http://arduino.cc/en/Main/ArduinoBoardUno`

[ATmega328p] Atmel. ATmega328P datasheet. `http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega328_P%20AVR%20MCU%20with%20picoPower%20Technology%20Data%20Sheet%2040001984A.pdf`

[Instr] Joc d'instruccions dels AVR - `http://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf`

[AS] Manual de referència del Assemblador del GNU - `http://sourceware.org/binutils/docs/as/`

[ihex] Format de fitxer intel HEX - `http://en.wikipedia.org/wiki/Intel_HEX`

[Hardvard] Arquitectura de tipus Harvard - `http://en.wikipedia.org/wiki/Harvard_architecture`

[Endian] Ordre de disposició dels bytes - `http://en.wikipedia.org/wiki/Endianness`