Enginyeria de Sistemes TIC

Dispositius Programables

Pràctica 2 - Primeres proves

Francisco del Águila López

Setembre 2012
Escola Politècnica Superior d'Enginyeria de Manresa
Universitat politècnica de Catalunya

1 Objectiu

L'objectiu d'aquesta segona sessió és familiaritzar-se amb la utilització de les eines així com la definició dels primers programes en Assemblador.

2 Introducció teòrica

L'aplicació *avrdude* es fa servir per transferir els fitxers binaris cap a cadascuna de les possibles memòries del dispositiu. En general si el nostre fitxer binari prové de l'assemblat d'un programa i per tant és un codi binari executable, la memòria a la que transferirem aquest fitxer serà la memòria de programa del microcontrolador.

Les diferents memòries a les que podem accedir des del *avrdude* són:

flash: És la memòria en la que queda emmagatzemat el nostre programa, per tant és la memòria que típicament farem servir.

eeprom: És una memòria extra que tenen els AVR. A aquesta memòria no es pot accedir directament a través del bus d'adreces sinó que s'ha d'accedir mitjançant accessos a registres d'entrada sortida. Típicament és on es pot guardar un gran volum de dades de forma no volàtil.

hfuse lfuse efuse ... La resta de possibles memòries són petites zones reservades en els AVR per poder configurar paràmetres especials, per exemples l'identificador de dispositiu, certes configuracions, etc.

Els diferents tipus de memòria que es poden tractar es poden consultar al manual *man avrdude* en un terminal de Linux buscant la opció *-U*. En la descripció d'aquesta opció també es troben els formats de fitxers que admet l'avrdude.

Els formats dels fitxers poden ser:

r Format binari directe amb bytes ordenats de format little-endian [Endian]

i Intel Hex [ihex]

d Format de sortida (no es pot fer servir per escriure a la memòria, tan sols per llegir) decimal, separat per comes

h Format de sortida hexadecimal, cada valor va precedit de 0x

b Format de sortida binari, cada valor va precedit de 0b.

2.1 Els fitxers amb format ihex

El format Intel Hex és un format de text, i per tant visualitzable directament, on cada línia conté valors hexadecimals que codifiquen les dades i la seva posició d'adreça. A la referència[ihex] queda descrit aquest format.

2.1.1 Ordre de disposició dels Bytes

Per llegir o escriure totalment una memòria es segueix l'ordre determinat per les adreces, començant des de la posición zero fins al final. En el cas de memòries disposades en paraules de 8 bits (1 Byte), això no comporta cap complicació afegida. Però en el cas de memòries amb amples més grans a 1 Byte, quan accedim a dades d'amplada més petita, s'ha de determinar en quin ordre es fa la operació de lectura o escriptura. Aquesta disposició és el que determina l'*Endiannes* de la arquitectura de la memoria [Endian].

En l'arquitectura AVR aquest ordre és de tipus *Little Endian*. Això vol dir que quan volem omplir una memòria (ja sigui de dades o de programa) omplirem les posicions de dreta a esquerra.

En el cas de la memòria de programa, les paraules són de 16 bits (2 Bytes) per tant la matriu de memòria segueix l'ordre indicat per la següent taula:

1	0
3	2
5	4

2.2 El mapa de memòria dels AVR

La arquitectura dels AVR és de tipus harvard [Hardvard]. La descripció detallada d'aquesta arquitectura i d'altres es veurà a les classes de teoria. Essencialment la arquitectura

Hardvard es caracteritza per tenir separats, i per tant en busos diferents, els diferents tipus de memòria. La existència en paralel de diferents busos permet que es puguin realitzar transferències simultànies que comporten accesos a les diferents memòries. D'aquesta manera, per exemple, es pot llegir la següent instrucció a executar, al mateix temps que es pot llegir una dada de la memòria de dades.

En general hi ha tres tipus de memòries que són les que queden separades:

Memòria de programa És l'espai reservat a contenir el codi executable. Està disposada en paraules de 16 bits, per tant 2 Bytes.

Memòria de dades Aquí trobem l'espai disponible per emmagatzemar les dades. Està disposada en paraules de 8 bits, per tant 1 Byte.

Memòria d'entrada / sortida És l'espai on es troben els registres dels dispositius que configuren la comunicació amb l'exterior i per tant els perifèrics del microcontrolador. Amb un ample també de 8 bits.

En el cas particular dels AVR la memòria d'entrada / sortida i la de dades tenen el mateix bus, per tant no estan separades. La memòria de programa sí que es troba totalment aillada de la de dades.

La memòria d'entrada / sortida es troba solapada sobre la memòria de dades. D'aquesta manera poden compartir el mateix bus de dades. Aquest solapament consisteix en que la memòria d'entrada / sortida comença a solapar-se a la posició 32 de la memòria de dades (veure apartat 8.3 - pàg 19 de [ATmega328p]). Aquestes 32 primeres possicions són les que estan reservades pels registres privilegiats de propòsit general.

El solapament de la memòria d'entrada / sortida implica que podem accedir a ella per dues vies diferents:

1. Una de les vies és fer servir les instruccions de codi màquina específiques per les entrades / sortides (**in**, **out**), però en aquest cas s'ha de tenir en compte que l'adreçament comença a comptar des de la posició 0.
2. L'altra via és fer servir les instruccions generals d'accés a memòria de dades (**ld**, **lds**, ...) però en aquest cas hem de tenir en compte que per accedir a la primera posició d'entrada / sortida s'ha de fer amb l'adreça 32.

Un llistat dels registres d'entrada / sortida es troba al capítol 31 - pàg 533 de [ATmega328p].

2.2.1 Registres de propòsit general

Les 32 primeres posicions de la memòria de dades la formen uns registres especials. Aquests registres tenen el privilegi de poder ser tractats amb un conjunt elevat d'instruccions d'Assemblador que altres posicions del mapa de memòria no poden (veure apartat 7.4 - pàg 12 de [ATmega328p]). Alguns d'ells encara són més especials pel fet de poder realitzar operacions amb dades de 16 bits.

2.3 Algunes de les instruccions en Assemblador

Pel desenvolupament d'aquesta pràctica cal el conèixement previ d'algunes de les instruccions dels AVR. La descripció detallada del joc d'instruccions es troba a [Instr].

2.4 Algunes directives de l'Assemblador de GNU

Les directives d'Assemblador permeten definir aspectes característics de l'assemblador que estem fent servir. el conjunt format per les directives i el joc d'instruccions de cada cpu és el que forma un programa en assemblador preparat per poder ser assemblat.

El manual de l'Assemblador del GNU on podem trobar la definició de les directives que es poden fer servir està a [AS]. Alguna d'aquestes directives són:

.global Necessita com a paràmetre un símbol. Fa que aquest símbol pugui ser visible pel linkador. A efectes pràctics cal definir l'etiqueta *main:* que indica l'inici del programa i s'ha de definir com a *global*.

.equ .set = Aquestes tres diretives són igual i permeten associar a un símbol una expressió qualsevol.

3 El primer programa

El primer programa més simple que podem fer podria ser:

```
.global main

main:
        nop
        rjmp main
```

En aquest primer programa és el contingut d'un fitxer de text anomenat *primer.S*. S'observa la definició de l'etiqueta d'inici de programa *main:* i la utilització de la directiva **.global**.

Un programa una mica més elaborat podria ser:

```
.global main

/* Això és un comentari
de més d'una línia */

/* Aquí es fan algunes definicions */
mregistre = 16
.set DDRB_o , 0x4
.equ PORTB_o , 0x5
```

```
/* Comença el programa principal */
main:
        ldi mregistre,0xFF
        out DDRB_o, mregistre
loop:
        ldi mregistre,0x00
        out PORTB_o, mregistre
        ldi mregistre,0xFF
        out PORTB_o, mregistre
        rjmp loop
```

Aquest programa permet que alternativament s'encengui i s'apagui el led de la placa Arduino.

3.1 Diferents formats numèrics per enters

La manera de definir nombres enters en l'assemblador del GNU "as" es troba a l'apartat 3.6.2.1 de [AS].

El resum és:

	Prefix	**Exemple**
binari	0b	0b11001
octal	0	07430
decimal		1395
hexadecimal	0x	0xF30C91

4 Estudi previ

1. Del següent fitxer en format Intel Hex treu l'adreça on estan les dades i numera en format hexadecimal els bytes que defineixen aquestes dades. No tingueu en compte el camp de checksum.

   ```
   :040000000000000C0XX
   :00000001FF
   ```

2. Considera els bytes extrets del fitxer anterior com els opcodes d'instruccions màquina dels AVR. Dedueix quines instruccions màquina corresponen, tenint en compte que la disposició de memòria és del tipus *Little Endian*, buscant entre les següents instruccions: **nop**, **rjmp**, **ldi**, **out**, **subi**, **brne**.

3. Considera el primer exemple de programa en Assemblador. Extreu els bits corresponents als opcodes d'aquestes instruccions i escriu-los en un fitxer en format Intel Hex sense tenir en compte el checksum. Recordeu de fer ús de la disposició de tipus Little Endian.
4. Tenint en compte que la memòria d'entrada / sortida està solapada a sobre de la memòria de dades. Quin sentit pot tenir definir instruccions específiques d'entrada / sortida si amb les instruccions generals d'accés a la memòria de dades podem fer el mateix.
5. Busqueu a [Instr] què fan i quins paràmetres fan servir les instruccions de llenguatje Assemblador: **nop**, **rjmp**, **ldi**, **out**, **subi**, **brne**.
6. Busqueu a [AS] quin significat tenen les directives **.global**, **.set**, **.equ**.
7. Localitza el pin de la plataforma Arduino que està connectat al led. Localitza quina pota de l'Arduino està connectada al led que existeix a la placa. Comprova a quin registre del mapa d'entrada / sortida correspont la pota de l'Arduino. Quin bit d'aquest registre està connectat al led? Quina és l'adreça del port que permet la modificació del estat del led (des del punt de vista del mapa d'entrada / sortida)? Quina és l'adreça del port que permet la modificació del estat del led (des del punt de vista del mapa de memòria)? La instrucció **out** serviria per modificar el valor del registre? La instrucció **ldi** serviria per modificar el valor del registre? Quina seria la sintaxis d'aquestes ordres (en el cas de ser apropiades) per encendre el led?
8. Tenint en compte la quantitat de cicles de rellotje que cal per executar les instruccions **ldi**, **out**, **rjmp**, Calcula a quina freqüència conmutarà el led de la placa del segon programa d'exemple. Està la mateixa estona encès que apagat?
9. Amplia el codi del segon exemple amb la intenció que la freqüencia de de conmutació sigui la meitat i que estigui la mateixa estona encès i apagat. Nombra aquest fitxer de codi font com *tercer.S*.
10. Modifica el codi per crear el fitxer quart.S per tal que la commutació del led pugui ser visible per l'ull. Això implica que la freqüencia de commutació ha de ser inferior a uns 20 Hz aproximadament. L'objectiu és entretenir el microcontrolador abans de que faci cada canvi d'estat del led.

5 Treball pràctic

A proposta del professor es farà diverses proves amb el Toolchain de GNU començant per programes molt senzills. Es veurà l'estructura bàsica d'un programa i s'acabarà la pràctica amb la proposta d'encendre i apagar el led de la placa Arduino.

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, és a dir, *prova.S* és el fitxer d'entrada i *prova.elf* és el fitxer de sortida generat.

Un paràmetre que pot ser d'utilitat és **-E**, en aquest cas es realitza l'asssemblat estricatament. 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 desassemblat. En aquest cas, a partir d'un fitxer executable de tipus *elf* s'obté un fitxer en codi font. Obviament la informació extra que conté el codi font original ha desaparegut. La comanda és

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

5.2 Assemblat del codi font sense complements

El procés de l'apartat anterior és el procés normal que s'ha de fer quan es vol programar el microcontrolador. Aquest procés implica que l'assemblador afegeixi alguns extres en el codi per tal de deixar ben configurades altres funcionalitats de l'AVR com per exemple les interrupcions, l'apuntador de la pila, inicialització alguns registres, etc. Aquests extres provenen dels fitxers de *startup crt*.

Per evitar afegir en el codi executable aquesta inicialització i deixar estrictament el programa, la comanda és la següent

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

Les altres comandes per desassemblar o convertir al format *ihex* segueixen sent vàlides.

Recordeu que el manual d'aquestes comandes es troba a *man nom_comanda*.

5.3 Transferència dels fitxers executables (Avrdude)

Quan es connecta l'Arduino al ordinador pel 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'apareixer un nou dispositiu amb identificació **ttyACM0.**

Per comporvar 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 comunicions 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 contintgut seria

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

5.4 Tasques

1. Assembla el primer programa d'exemple i genera el fitxer executable amb nom *primer*. Escull les opcions d'assemblat per generar finalment un fitxer que s'executi correctament a l'AVR.
2. Converteix el fitxer *primer.elf* a *primer.hex*.
3. Visualitza el fitxer *primer.hex* i comprova si trobes el que esperes. Localitza els opcodes que s'han fet servir indicant on es troben en el fitxer.
4. Si la tasca 3 no ha tingut resultat satisfactori, realitza el procés invers de desassemblat del fitxer *primer.elf* cap a *primer.disam* i visualitza el seu contingut. És el mateix que el fitxer original en assemblador? Comenteu el resultat.
5. Repeteix els passos 1-4 amb les opcions d'assemblat estrictes corresponent al codi del programa (sense inicialitzacions extres).
6. Modifica el fitxer font afegint un parell de instruccions nop al començament. renombra el nou fitxer amb *primer2.S* i comprova els canvis produïts tant en el fitxer *elf, hex* i *disam*. Són els que toquen?
7. Llegeix el contingut de la memòria de programa i crea el fitxer *original.hex*. Visualitza el contingut i comenta'l.
8. Transfereix el contingut de *primer.hex* a la memòria de programa de l'Arduino. Podem comprovar si s'està executant correctament?
9. Assembla el segon programa d'exemple i genera el fitxer executable amb nom *segon*. Transfereix el programa a l'Arduino i comprova el resultat. Podem comprovar si s'està executant correctament? Mesura la freqüència a la que commuta el led i comprova que encaixa amb els càlculs de l'estudi previ.
10. Transfereix el programa *tercer.S* a l'Arduino i comprova si fa el que s'espera.
11. Transfereix el programa *quart.S* a l'Arduino i comprova si fa el que s'espera.
12. Proposeu alguna modificació o un nou programa *cinque.S* que us sembli interessant.

Referències

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

[ATmega328p] Atmel. ATmega328P datasheet. http://www.atmel.com/dyn/resources/prod_documents/doc8271.pdf

[AtmAss] Atmel Assembler documentation - http://www.atmel.com/atmel/acrobat/doc1022.pdf

[Instr] Joc d'instruccions dels AVR - http://www.atmel.com/atmel/acrobat/doc0856.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 Hardvard - http://en.wikipedia.org/wiki/Harvard_architecture

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