Enginyeria de Sistemes TIC

Dispositius Programables

Pràctica 8 - Màquina de xifrar

Francisco del Águila López

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

1 Objectiu

L'objectiu d'aquesta pràctica és realitzar una màquina per xifrar missatges de text. L'AVR rebrà pel port serie un missatge escrit en text i retronarà pel port serie el missatge xifrat. El sistema de xifrat està basat en una traducció caràter a caràcter, es a dir, byte a byte. L'algoritme de xifrat es basa en una taula d'equivalència on a cada caràcter de text en clar li correspon un caràcter de text xifrat.

2 Descripció de la màquina de xifrar

Aquest xifrador funciona a nivell de byte, es a dir, tansforma un byte en un altre. No hi ha un algoritme de transformació definit, com podria ser sumar (en mòdul 256) un desplazament al byte d'entrada per donar el byte de sortida. L'algoritme de transformació es basa en una taula on cada possible byte d'entrada està aparellat amb un altre byte de sortida.

Els possibles bytes del text en clar són el codi ASCII de les lletres de l'abecedari. L'entrada de dades tant en majúscules com en minúscules s'ha de tractar de la mateixa manera, per tant aquest detall s'ha de tenir en compte per definir el programa.

Els possibles bytes de text xifrat són caràcters imprimibles de la taula ASCII, per tant l'aplicació de comunicació pot ser sense problemes un terminal com és el *picocom*. També és possible fer servir l'aplicació *moserial*.

Per simplificar aquesta màquina, el caràcter corresponent al espai es deixarà sense trasnsformar, això donarà una idea de quina mida tenen les paraules xifrades, al mateix temps

que fa el sistema de xifrat més feble. Aquest sistema de xifrat és de tipus monoalfabètic per tant un atac estadístic amb un text xifrat prou llarg seria suficient per trencar-lo. De totes maneres es prou interessant ja que permet fer ús de recursos de l'AVR que no s'han vist fins ara.

3 Nous recursos

3.1 Recursos de l'AVR

Amb aquesta pràctica es farà ús de la memoria de dades propiament dita a la qual podem definir les variables que ens puguin interessar a part de disposar de la taula de xifrat. També s'utilitzarà l'adreçament indirecte a la memòria de dades.

Per mantenir la compatibilitat amb altres assembladors d'altres màquines i amb la intenció de que l'assemblador sigui genèric per moltes màquines, l'assemblador del GNU pot definir diferents seccions en el codi. Les més importants són:

text: És on es troba el programa propiament dit, per tant el seu contingut són els opcodes que formen el programa.

data: És on es troben dades que contenen valors concrets inicialitzats. Aquí és on trobarem, per exemple, la taula de xifrat.

bss: És on es troben dades inicialitzades amb valor 0. Aquí es trobaran les variables generals inicialitzades a 0.

L'aplicació particular d'aquestes seccions a l'arquitectura Harvard de l'AVR té diferents implicacions:

1. La secció *text* és directament el codi, de fet, tots els programes de les pràctiques fetes fins ara estan associats per defecte a la secció text.
2. La secció *data* conté variables que contenen valors concret. Aquests valors s'han de poder conservar per diferents posades en funcionament, per tant han d'estar guardats en una memòria no volàtil. Per altra banda, aquestes variables són dades, per tant han d'estar localitzades a la memòria de dades (SRAM) que és volàtil. Prescindint de la memòria especial EEPROM que no tots els dispositius AVR contenen, la manera de compatibilitzar aquest fet és que les variables estiguin en una zona de la memòria de programa de l'AVR i en el moment de l'arrencada del sistema es faci una còpia de la memòria de programa a la de dades. Aquest procés queda automatitzat en l'entorn del GNU GCC.
3. La secció *bss* conté variables de contenen valors a 0. En aquest cas no cal guardar cap valor en cap memòria no volàtil. Però sí que cal que en el moment de l'arrencada el sistema reservi l'espai d'aquestes variables a la memòria de dades i a més les fixi a valor 0. Aquest procediment també es fa de manera automàtica.

Aquestes tasques extres que s'han de realitzar en el moment d'arrencar el sistema que afecten a les seccions *bss* i *data* es porten a terme si en el nostre codi declarem uns símbols concrets i reservats com globals:

.global __do_copy_data Això permet que les dades que es troben a la secció *data* siguin copiades a la memòria de dades.

.global __do_clear_bss Això permet que es reservi l'espai a la memòria de dades per les valiables que estan inicialitzades a 0.

3.2 Recursos de l'assemblador

A continuació es descriuen algunes directives d'assemblador útils quan els programes comencen a ser complexos. La descripció detallada d'aquestes directives es troben a [AS].

.section Serveix per definir a quina secció va el codi que ve a continuació. Requereix com a paràmetre el nom de la secció.

.text .data .bss Defineixen diferents tipus de seccions. I per tant, defineixen a quina secció anirà el codi que ve a continuació.

.include Serveix per incloure just en aquell punt el contingut d'un fitxer extern que es farà servir en el programa.

.skip .space Aquestes directives són equivalents i serveixen per omplir un número de bytes definit pel primer paràmetre amb el contingut definit pel segon paràmetre. Els paràmetres poden ser expressions.

.fill També serveix per omplir dades amb valors concrets, similar a .skip o .space.

.byte Accepta expressions per omplir el resultat en forma de byte. La separació amb coma omple els bytes de forma consecutiva.

.ascii .asciiz Omple de forma consecutiva la memòria amb els caràcters del string que es dona com a paràmetre. LA opció asciiz acaba amb la última posició amb el byte 0x00.

.if .else .elseif .endif Serveixen per incloure un tros de codi o un altre en funció d'una condició. La forma més simple de definir una condició és evaluar una expressió basada en la definició d'un símbol i el seu valor. Al programa de mostra es pot veure un exemple.

3.2.1 Modificadors

Les adreces de memòria de dades en el ATmega328p són de 16 bits. Per tractar aquestes adreces amb els registres existents de 8 bits tenim la necessitat de diferenciar el byte de més pes i el byte de menys pes. L'assemblador del GNU disposa d'uns modificadors específics per tractar amb els AVR que permeten treballar fàcilment amb les adreces de 16 bits. Al programa d'exemple es veu com es pot fer ús d'aquests modificadors.

hi8() Permet extreure d'una etiqueta de posició de memòria (16 bits) el byte de major pes.

lo8() Permet extreure d'una etiqueta de posició de memòria (16 bits) el byte de menor pes.

4 Exemple de programa

L'exemple de codi proposat fa servir l'adreçament indirecte per accedir a una taula disposada en memòria per columnes. Aquesta taula és de 2 columnes i a cada pulsació de tecla (enviament pel port serie) fa un volcat de la primera columna fins la possició marcada pel símbol SALTA i continua el volcat amb la segona columna fins el final de la segona columna. El valor de SALTA queda fixat en temps de compilació.

La taula estableix la equivalència del xifrat. La primera columna defineix el text en clar i la segona columna defineix el text xifrat. Aquesta taula a estat inspirada en funció dels codis que dona un teclat PS2, modificant alguns d'ells per evitar els codis no imprimibles.

Per poder comprovar com queda la memòria de programa amb les diferents seccions executeu l'ordre

```
avr-objdump -s exemple8.ihex
```

un cop generat el fitxer Intel Hex.

5 Estudi previ

1. Desassembleu el codi executable *.elf* de exemple.S. Comproveu que el programa fa el que toca i descriviu els grans blocs que forma el desassemblat del codi. Heu trobat on és la taula de xifrat?
2. Executeu *avr-objdump -s exemple8.ihex* i descriviu els grans blocs de dades que s'obtenen. Localitzeu la part de codi executable, dades inicialitzades i dades no inicialitzades. Heu trobat totes les seccions?
3. Dissenya una programa que quan s'envia el caràcter "1" retorni el contingut de la posició de la primera columna definida pel símbol SALTA. I que quan s'envia el caràcter "2" retorni el contingut de la posició de la segona columna definida pel símbol SALTA.
4. Modifica el programa anterior perquè quan el símbol DEBUGAR està activat (valor =1), a cada pulsació retorni primer el caràcter que s'ha enviat i després el que demana l'apartat 1.
5. Modifica el programa anterior perquè vagi guardant en la variable VEGADES el número de caràcters rebuts des del moment de reset. Si el mode de DEBUGAR

està actiu serà el tercer caràcter retornat. Si el mode de DEBUGAR no està actiu només es retornarà l'únic byte demanat a 1.

6. Dissenya un programa que busqui a la primera columna la tecla pulsada. Si la troba, retorna una "S" i si no la troba retorna una "N"
7. Dissenya el programa xifrador. Recorda que ha d'admetre majúscules i minúscules i que el caràcter espai no l'ha de xifrar. Si s'entra qualsevol altra caràcter no corresponents a les lletres del abecedari no ha de retornar res.
8. *Opcional:* Dissenya el programa desxifrador basat en la operació contraria al xifrador.
9. *Opcional:* Dissenya un programa xifrador/desxifrador. Amb l'enviament del caràcter "0" està en mode xifrador i amb l'enviament del caràcter "9" està en mode desxifrador.

Podeu ampliar el vostre codi pels apartats 4, 5, 6 i 7 per admetre mode de DEBUGAR al vostre gust.

6 Treball pràctic

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

6.1 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
```

6.2 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'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
```

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`

[USART] Dispositiu USART `http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter`

[ASCII] Taula de caràcters ASCII `http://en.wikipedia.org/wiki/ASCII`