# Aritmética en coma flotante y programación en ensamblador en las prácticas de Estructura de Computadores: Control de un simulador de radar

P.A. Castillo Valdivieso, M. García Cruz, M.G. Arenas, G. Romero, A. Prieto Espinosa
Dpto. de Arquitectura y Tecnología de Computadores
Universidad de Granada, 18071 Granada
e-mail: pedro@atc.ugr.es

## Resumen

La enseñanza del lenguaje ensamblador es complicada debido a la dificultad que supone a los alumnos enfrentarse a un lenguaje de bajo nivel.

La experiencia nos dice que plantear unas prácticas atractivas al alumno les hace interesarse por dicho lenguaje y por la asignatura de Estructura de los Computadores I (en la que se estudia, en la Universidad de Granada).

La programación de un control de radar puede ser una práctica atractiva a los alumnos, de forma que la programación en ensamblador resulte casi un juego conforme mejoran la programación del cálculo de las coordenadas.

## 1. Motivación

La asignatura de Estructura de los Computadores I, en la Universidad de Granada, estudia el lenguaje máquina, centrándose en la arquitectura de la familia 80x86. Las prácticas de esta asignatura se basan, principalmente, en el estudio del lenguaje ensamblador.

A pesar de las ventajas que presenta este lenguaje en cuanto a poder realizar programas completamente adaptados a la máquina y por tanto optimizados en cuanto a velocidad y utilización de recursos [1,2], su estudio debe plantearse de forma que sea lo más atractivo y cómodo para el alumno.

En este trabajo se presenta una herramienta software que simula un radar que debe ser controlado mediante un programa externo que calcule una coordenada a partir de un ángulo y una distancia. Esta práctica puede resultar atractiva a los alumnos, de forma que la programación en ensamblador sea más amena.

El radar realiza un barrido de 360 grados detectando de cuando en cuando, de forma aleatoria, objetivos que debemos capturar.

El programa intercepta un objetivo a partir de la coordenada que le transmita nuestro programa de control externo. Para ello, ambos programas se sincronizan a través de dos ficheros de texto: el simulador escribe el valor del ángulo y la distancia al objeto detectado, y nuestro programa escribe la coordenada de intercepción.

Además de la implementación de los accesos a los ficheros y de la sincronización de los programas, se deben realizar cálculos matemáticos en coma flotante para el cálculo de funciones trigonométricas. Para la práctica, el programa de control se debe desarrollar en lenguaje ensamblador, con objeto de que aprendan dicho lenguaje, y la arquitectura 80x86.

El simulador ha sido desarrollado en C++, utilizando la biblioteca de programación gráfica ***gtkmm2*** [3], de forma que puede ser compilado y utilizado fácilmente, tanto en Windows como en Linux/Unix.

El resto del artículo está estructurado como sigue: en la siguiente sección se describe el simulador desarrollado; en la tercera sección se explica cómo usar el simulador en prácticas. La cuarta sección expone diferentes formas de calcular el seno y coseno para el cálculo de las coordenadas a partir de la distancia y ángulo. Por último, la quinta sección expone una serie de conclusiones.

## 2. El simulador

El sistema constará de dos programas (ver las Figuras 1 y 3): un simulador que muestra el radar (programa gráfico, a la izquierda), y otro programa que realiza el control del primero

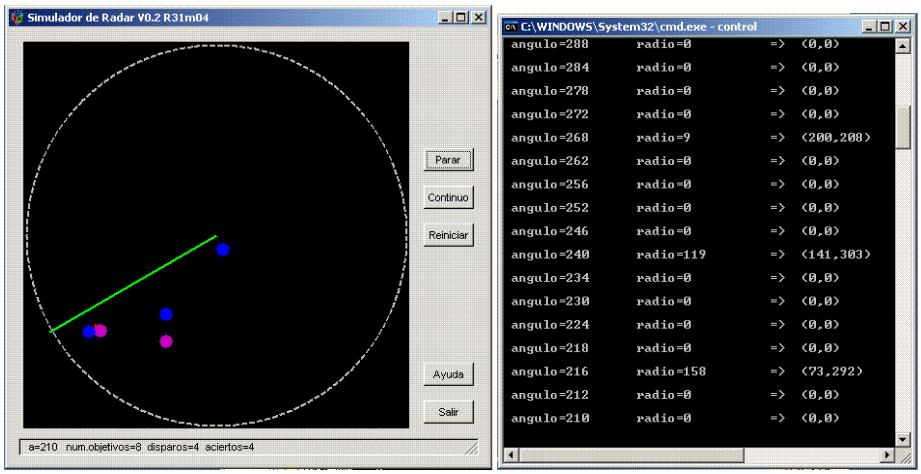


Figura 1. Captura de pantalla bajo Windows en la que el programa-control (a la derecha) envía coordenadas al simulador del radar (a la izquierda). En este caso, el radar muestra cinco objetivos, de los cuales el control ha interceptado a dos (a la derecha vemos las coordenadas a las que se ordena interceptar). Conforme el detector del radar avanza, los objetivos que quedan más lejanos van desapareciendo.

(programa que se ejecuta en modo texto en una ventana de comandos, a la derecha).

El sistema funciona de la siguiente forma:

1. La sincronización entre el programa-control y el simulador se hace a través de dos ficheros (S y A).
2. El simulador inicializa la ventana, dibuja el radar, y va haciendo avanzar el indicador de forma constante. A cada paso escribe en el fichero S el ángulo actual del indicador y la distancia a un objetivo detectado en ese momento (si no hubiera objetivo, escribe el valor 0 como distancia).
3. El programa-control, que estaba en un bucle de espera hasta que hubiera datos en el fichero S, lee los 2 valores. Se trata de acceder al fichero en modo de texto e ir leyendo carácter a carácter hasta encontrar un espacio, lo que indicará que se ha leído un número más. Seguiremos leyendo esa línea hasta haber leído los 2 números; después se pasan esas cadenas leídas a números enteros.
4. El control procesa esos dos valores para calcular la coordenada cartesiana (X,Y) donde debemos interceptar. Es posible que en alguna ocasión (si la distancia es 0, esto es, si no hay objetivo), el control decida no actuar; en ese caso, los valores para X e Y deben ser igual a 0. En cualquier otro caso, las ecuaciones a utilizar serán las siguientes:
   X = 200 + distancia * cos(angulo)
   Y = 200 - distancia * sen(angulo)
5. Una vez calculada la coordenada a interceptar, se guardan los valores X e Y en el fichero A (en la primera línea, separados ambos números por un espacio). Para ello hay que pasar a cadena cada uno de los números, y escribirlos en el fichero como caracteres individuales (texto plano). Así, si la coordenada es (45,70) se guardaría como la secuencia de caracteres '4', '5', ' ', '7', '0', 13, 10 (el \n , en MSDOS, se cambia por los dos caracteres con códigos ASCII 13 y 10).

La sincronización por ficheros hace más fácilmente portable el sistema, y al tiempo delimita los aspectos que el alumno debe estudiar para hacer la práctica:

- E/S por pantalla
- E/S por fichero
- Estructuras de control complejas para la sincronización

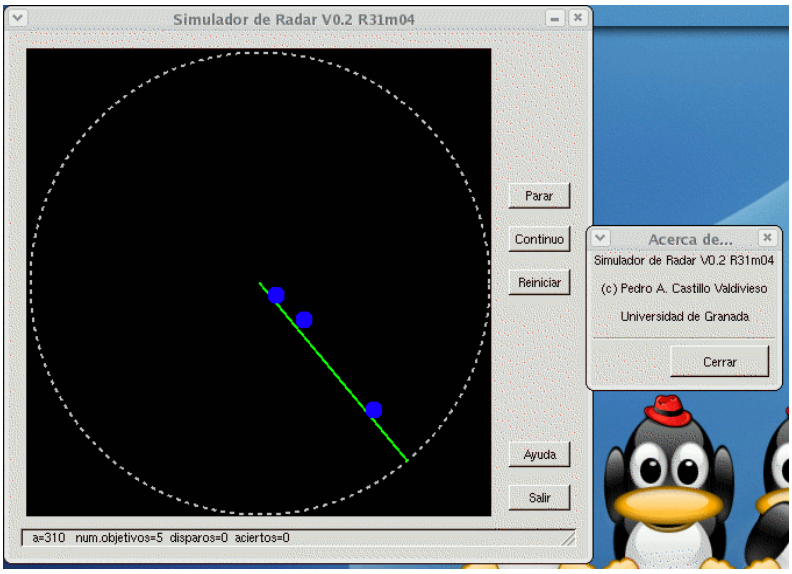


Figura 2. Controles del simulador para iniciar o detener su funcionamiento, para reiniciarlo, obtener información sobre la versión del programa, o para terminar su ejecución.

- Cálculos matemáticos con números enteros y en coma flotante
- Formato IEEE-754 de representación de números reales
- Transformación de datos entre diversas representaciones

## 3. ¿Cómo usar el simulador en prácticas?

Hemos desarrollado una versión del simulador para Linux y otra para Win32, por lo que la práctica puede plantearse en ambos sistemas operativos.

Las versiones del simulador para los diferentes sistemas pueden descargarse de la URL *http://atc.ugr.es/~pedro/docencia/simuladores*

El programa simulador dispone de varios botones para iniciar o detener su funcionamiento, o reiniciarlo (ver Figura 2). Su función queda descrita a continuación:

- "***Parar***" : detiene el simulador. El indicador deja de girar y las comunicaciones se paran.
- "***Continuo***" : comienza el funcionamiento del radar. El indicador irá girando, y de cuando en cuando detectará nuevos objetivos. Las comunicaciones se llevarán a cabo continuamente con el programa de control.
- "***Reiniciar***" : si queremos probar diferentes programas-control, pulsaremos este botón, de forma que se reinicie el valor del número de objetivos y aciertos, y el indicador se sitúe en su posición inicial.
- "***Ayuda***" : muestra información sobre la versión del programa simulador.
- "***Salir***" : termina el programa.

Para sincronizar ambos programas a través de los dos ficheros, el bucle principal del programa-control (el programa que debemos realizar en ensamblador) debe ser similar al siguiente:

```
repetir
  abrir_fichero("S")
  mientras(no_hay_datos_en("S")){
    cerrar_fichero
    hacer_pausa
    abrir_fichero("S")
   fin_mientras
  // leer los datos
  angulo    = leer_del_fichero
  distancia = leer_del_fichero
  cerrar_fichero
  //calcular coordenada
  x = calcular_coord_X
  y = calcular_coord_Y
  //escribir en "A" la coordenada
```

```
    abrir_fichero("A")
    escribir_en_fichero x , y
    cerrar_fichero
fin_repetir
```

## 4. Cálculo del seno y coseno

Para realizar esta práctica es imprescindible utilizar números reales para el cálculo del seno y del coseno de un ángulo (ver el Apéndice I para una descripción de los cálculos en coma flotante usando el coprocesador matemático). En cuanto a posibles formas de calcular el seno y coseno de un ángulo, daremos varias posibilidades.

### 4.1. Desarrollo en serie

Podemos calcular el seno y coseno de un ángulo mediante los siguientes desarrollos en serie:

$$sen(x) = \sum_{n=1}^{\infty} (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!}$$

$$\cos(x) = \sum_{n=1}^{\infty} (-1)^{n-1} \frac{x^{2n-2}}{(2n-2)!}$$

Las sumatorias las haremos con **t** términos, tal que la diferencia entre la suma desde **1 hasta t** y la suma desde **1 hasta t+1** sea menor que una constante dada que determinará la precisión.

### 4.2. "Geometría recreativa"

La idea básica la tomamos de la página web: *http://es.geocities.com/geometriarecreativa/geomrecreat05.html*
En ella se nos explica cómo calcular ciertas funciones matemáticas de una forma un tanto "manual", sin recurrir a tablas, calculadoras, ni nada.

Es cierto que con estos cálculos, el error cometido puede variar y ser algo más alto que con el método anterior, pero resulta mucho más sencillo de programar.

La idea básica que se expone es subdividir el primer cuadrante del círculo para ir calculando el seno de cada ángulo simplemente sumando una cantidad fija que podemos calcular por semejanza de triángulos. Según se explica en la página, podemos cometer errores de menos del 2% en el cálculo del seno de cualquier ángulo.

Pues bien, nosotros vamos a partir de los valores conocidos para ciertos ángulos, y vamos a obviar el porcentaje de error cometido para calcular los ángulos intermedios:

| | |
|---|---|
| Sen(0) | 0 |
| Sen(15) | 0,259 |
| Sen(30) | 0.5 |
| Sen(45) | 0.707 |
| Sen(60) | 0,866 |
| Sen(75) | 0,966 |
| Sen(90) | 1 |

Para ángulos intermedios sumaremos un valor calculado como *(sen(a2)-sen(a1))/15* de forma que obtendremos un valor aproximado con un error aceptablemente pequeño.

De esta forma, y a partir de estos cálculos podemos obtener los valores correspondientes al seno de los ángulos de los restantes cuadrantes, y los correspondientes al coseno.

### 4.3. Construir la tabla completa

Una variación consiste en tener una tabla completa (360 entradas, para los 360º) para el cálculo del seno y otra diferente para el cálculo del coseno. De esta forma, para cada valor del ángulo sólo tenemos que ir a la posición de la tabla para obtener el valor ya calculado.

En este trabajo no las vamos a poner, y de hecho se recomienda utilizar cualquiera de los otros métodos antes que éste.

### 4.4. Usar las instrucciones del coprocesador

La más rápida y sencilla de las opciones propuestas. El Apéndice I describe la forma de realizar cálculos en coma flotante usando el coprocesador matemático. Las instrucciones aritméticas que utilizaremos son las siguientes:

**FCOS** : calcula el coseno del valor contenido en la cima de la pila del coprocesador. Se supone expresado en radianes (habrá que transformar los ángulos leídos en grados a radianes, multiplicando por PI y dividiendo por 180). Se sustituye la cima de la pila por el valor calculado.

**FSIN** : calcula el seno del número real contenido en la cima de la pila. Se supone expresado en radianes. Se sustituye la cima por el valor obtenido.

**FSINCOS** : halla el seno y el coseno del ángulo (en radianes) que hay en la cima de la pila, y deja ambos valores en la cima y en la siguiente posición de la pila.

## 5. Conclusiones

El aprendizaje del lenguaje ensamblador suele resultar complicado a los alumnos, así como llegar a hacerles ver la necesidad de su estudio. Sin embargo, plantearles unas prácticas amenas y atractivas puede motivarlos, evitando que les resulte tan engorroso el estudio a tan bajo nivel de la programación.

Este trabajo presenta un simulador de radar desarrollado con esta idea. El alumno debe desarrollar un programa de control que recibe el valor de la distancia y ángulo del objetivo detectado, y le envía la coordenada cartesiana del objetivo a interceptar.

El primer año en que se ha ofertado esta práctica varios alumnos han desarrollado programas de control siguiendo las diferentes opciones para el cálculo del seno y coseno. Les resultó estimulante y motivador comprobar cómo sus programas calculaban las coordenadas correctamente y acertaban en los objetivos.

## Referencias

[1] RODRÍGUEZ-ROSELLÓ, M. A.: "8088-8086/8087. Programación ensamblador en entorno MSDOS". Anaya Multimedia, 1993

[2] BELTRÁN DE HEREDIA, Jon: "Lenguaje ensamblador de los 80x86" Anaya Multimedia. Colección Guías prácticas para programadores. ISBN: 84-7614-622-1

[3] http://gtkmm.sourceforge.net/gtkmm2

[4] STEVENSON, D.: "A Proposed Standard for Binary Floating-Point Arithmetic: Draft 8.0 of IEEE Task P754." IEEE Comput. 14 51-62, 1981.

[5] A. Prieto Espinosa, A. Lloris Ruiz, J.C. Torres Cantero: "INTRODUCCIóN A LA INFORMáTICA 3ªed". Editorial McGraw-Hill. ISBN: 8448132173

## Apéndice I. Arquitectura del coprocesador

El funcionamiento de los registros internos del coprocesador matemático es diferente al del 8086. No existen aislados ni con nombres distintos. Están estructurados en forma de pila cíclica de ocho elementos. Cada elemento de la pila es de 10 bytes de longitud (80 bits), y el formato de los datos es real temporal, de forma que el rango de valores representable va de 3.4E-4932 hasta 1.2E4932. La representación binaria interna sigue el formato IEEE754 [4,5].

El puntero de pila indica en todo momento cuál es el elemento que se encuentra en la cabecera de la pila (**S**tack **T**op). Puede valer entre 0 y 7, es decir, señala el número del elemento.

La mayor parte de las instrucciones del 8087 usan los registros de la pila. Así, las aritméticas utilizan uno o dos operandos de la pila y el resultado se deja también en la pila.

Todos los nombres simbólicos de las instrucciones empiezan por F. El conjunto de instrucciones se puede dividir en los siguientes tipos(ver [1,2] para obtener más detalles):

- de transferencia de datos. Sirven para mover operandos entre los elementos de la pila y la memoria:

  | | |
  |---|---|
  | FLD | cargar real en la pila |
  | FST | almacenar real en mem. |
  | FILD | cargar entero en la pila |
  | FIST | almacenar entero en mem. |

- Aritméticas. Permiten hacer operaciones aritméticas. Los operandos pueden ser elementos de la pila o de memoria. El resultado se almacena en la pila:

  | | |
  |---|---|
  | FADD | sumar reales |
  | FIADD | sumar num. enteros |
  | FSUB | restar num. reales |
  | FISUB | restar num. enteros |
  | FMUL | multiplicar reales |
  | FIMUL | multiplicar enteros |
  | FDIV | dividir reales |
  | FIDIV | dividir enteros |
  | FSQRT | raíz cuadrada |
  | FABS | valor absoluto |

- de comparación. Sirven para comparar dos operandos. Para analizar el contenido del código de condición hay que transferir a memoria la palabra de estado:

  | | |
  |---|---|
  | FCOM | comparar dos valores reales |
  | FICOM | comparar dos valores enteros |

FXAM examina el tope de la pila, para determinar si contiene un valor normalizado, renormalizado, cero, infinito o vacío.

- de cálculo de funciones. Sirven para calcular funciones trigonométricas, logarítmicas y exponenciales:
  FPTAN tangente de un ángulo
  FCOS coseno de un ángulo
  FSIN seno de un ángulo
  FSINCOS seno y coseno de un ángulo

- relativas a constantes. Sirven para cargar una constante en la pila:
  FLDZ cargar 0.0
  FLD1 cargar 1.0
  FLDPI cargar PI

- de control del microprocesador. Corresponden a actividades internas del coprocesador (inicialización, cambiar modo de redondeo, permitir o inhibir interrupciones, etc):
  FINIT inicializar coprocesador
  FDISI desactivar ints
  FENI activar ints
  FSTSW almacenar palabra de estado
  FNOP no operación

Las del cálculo del seno y coseno corresponden al 80387, de forma que el 8087 no puede ejecutarlas.

A modo de ejemplo, veamos cómo realizar tres operaciones aritméticas sencillas y la comparación entre dos valores. No mostraremos todo el programa ensamblador, sino sólo las instrucciones que usaríamos para hacer el cálculo.

**Z = Y – X * X**

```
fld dword [x]          ; st0=x

; st0=st0*st0
fmul st0               ; st0=x*x     st1=x

fld dword [y]          ; st0=y     st1=x*x     st2=x

; st0=st1-st0
fsubrp st1             ; st0=y-x*x    st1=y   st2=x*x

fistp dword [z]        ; z=st0
```

**Z = sqrt( (X-Y) / Z )**

```
fld dword [y]          ; st0=y
fld dword [x]          ; st0=x      st1=y

fsubrp st1             ; st0=x-y    st1=x     st2=y

fld dword [z]          ; st0=z      st1=x-y    st2=x

fdivp st1              ; st0=(x-y)/z    st1=z    . . .

fsqrt                  ; st0=sqrt((x-y)/z)    . . .

fistp dword [Z]        ; Z=st0
```

**X = dist * cos(angulo)**

```
fld dword [pidiv180]   ; st0=PI/180
fld dword [ang]        ; st0=ang      st1=PI/180

fmul st1               ; st0=a=ang*PI/180

fcos                   ; st0=cos(a)     st1=a

fld dword [dist]       ; st0=dist     st1=cos(a)
fmul st1               ; st0=dist*cos(a)    st1=dist

fistp dword [X]        ; X=st0
```

**Comparar X e Y**

```
fldz                   ; st0= 0
fld dword [y]          ; st0=y     st1=0
fld dword [x]          ; st0=x     st1=y    st2=0

; comparer st0 con st1  (st0-st1)
fcomp st1
;cargar la palabra de estado en AX
fnstsw

; si (st0<st1) entonces ah=???????1   (x<y)
; si (st0>st1) entonces ah=???????0   (x>y)
and ah,00000001b
jnz   x_menor
```

```
        …
x_menor:
```

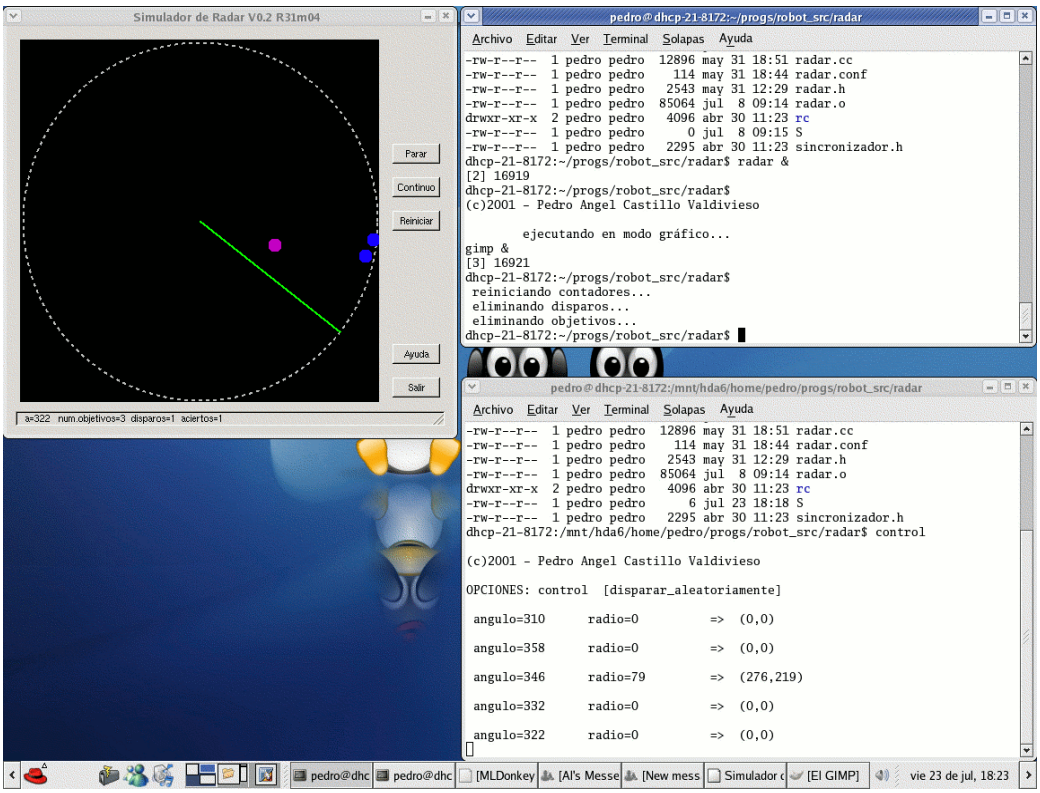


Figura 3. Captura de pantalla bajo Linux en la que el programa-control (abajo a la derecha) envía coordenadas al simulador del radar (arriba a la izquierda). En este caso, la pantalla del simulador muestra tres objetivos, uno de los cuales ha sido interceptado.