UNIVERSIDAD NACIONAL DE CORDOBA

FACULTAD DE CIENCIAS EXACTAS, FISICAS Y NATURALES

INGENIERIA EN COMPUTACIÓN

CATEDRA DE SISTEMAS DE COMPUTACIÓN

TRABAJO FINAL

Simulador de FPU 8087

Proyecto preliminar

versión 0.1

20 de mayo de 2009

Gaitán, Martín Emilio
leg. 29224785
Saffe, Jorge Nicolás
leg 32689033

Introducción

Se pretende continuar el trabajo realizado por Leonardo Rocha en 2008 como trabajo final de asignatura, el proyecto fpu8087sim, un Simulador de FPU multiplataforma, planteando un cambio de paradigma en el diseño del software: las intrucciones FPU se ejecutarán a nivel ensamblador en vez de simularse en alto nivel como en la versión actual.

Dicho proyecto ha sido liberado por el autor bajo Licencia Libre GPL3 y contamos con su beneplacito en la continuación del proyecto.

Descripción del proyecto actual

El software es un *simulador* de FPU programado Python, un lenguaje de alto nivel multiparadigma y multiplataforma. Está integramente basado en alto nivel, con un diseño orientado a objetos. Actualmente es capaz de simular unas 15 instrucciones de punto flotante.
fpu8087sim cuenta con una interfaz gráfica de usuario (GUI) compuesta por ventanas independientes: consola de entrada, registro de control, registro de estados, pila de FPU y consola de salida.
El software es capaz de ejecutar interactivamente intrucciones ensablador de FPU simples, una por vez.
El código está muy bien documentado, y cuenta con un diseño bastante robusto, donde se han marcado las líneas de continuación.

Descripción del trabajo a realizar

Como cambio fundamental de diseño, se pretende que la intrucciones se ejecuten **realmente** a nivel ensamblador. Esto implica realizar una doble interfaz bidireccional entre el lenguaje de alto nivel actual en Python a una capa middleware en Lenguaje C que interactua con las distintas rutinas de ensamblador. En todo momento, se llevará una copia de la pila de FPU en una estructura de datos de alto nivel.
Se pretende a la vez mejorar la interfaz de usuario, unificando las ventanas independientes en un solo entorno de trabajo donde el usuario pueda interactuar con el simulador y ver el estado global de los distintos componentes en juego sin cambiar de ventana.
Asimismo se desea lograr que el software sea multiplataforma, permitiendo ejecutarlo tanto en sistemas Windows como en Linux.
Por último, se pretende ampliar el set de instrucciones ejecutables desde el programa.

Marco teórico

El coprocesador matemático 8087

El **8087** fue el primer coprocesador numérico diseñado por Intel y fue construido para ser apareado con los microprocesadores Intel 8088 y 8086. El propósito del 8087 fue acelerar los cómputos en aplicaciones exigentes que implicaban cálculos con punto flotante. Este coprocesador introdujo cerca de 60 instrucciones nuevas disponibles para el programador.
La familia x87, dispone de registros estructurados en forma de stack con un rango desde ST0 a ST7.

Con el 80486 el coprocesador matemático se integró al chip del microprocesador.

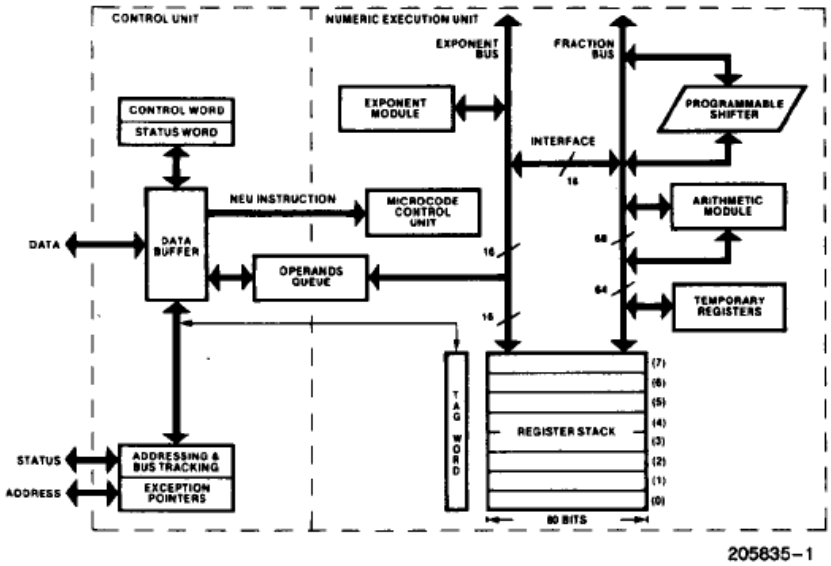


Figure 1. 8087 Block Diagram

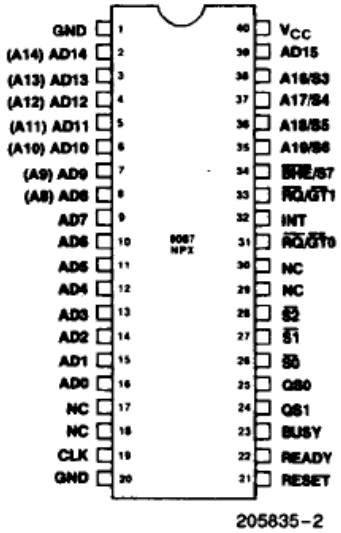


Figure 2. 8087 Pin Configuration

Illustration 1: Arquitectura del FPU

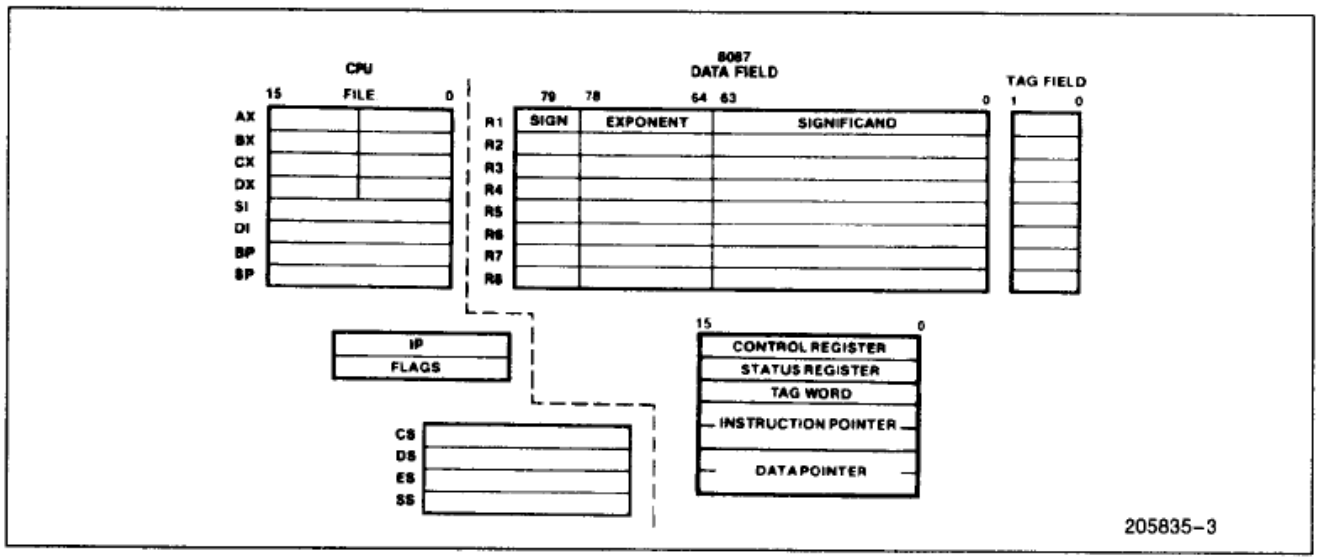
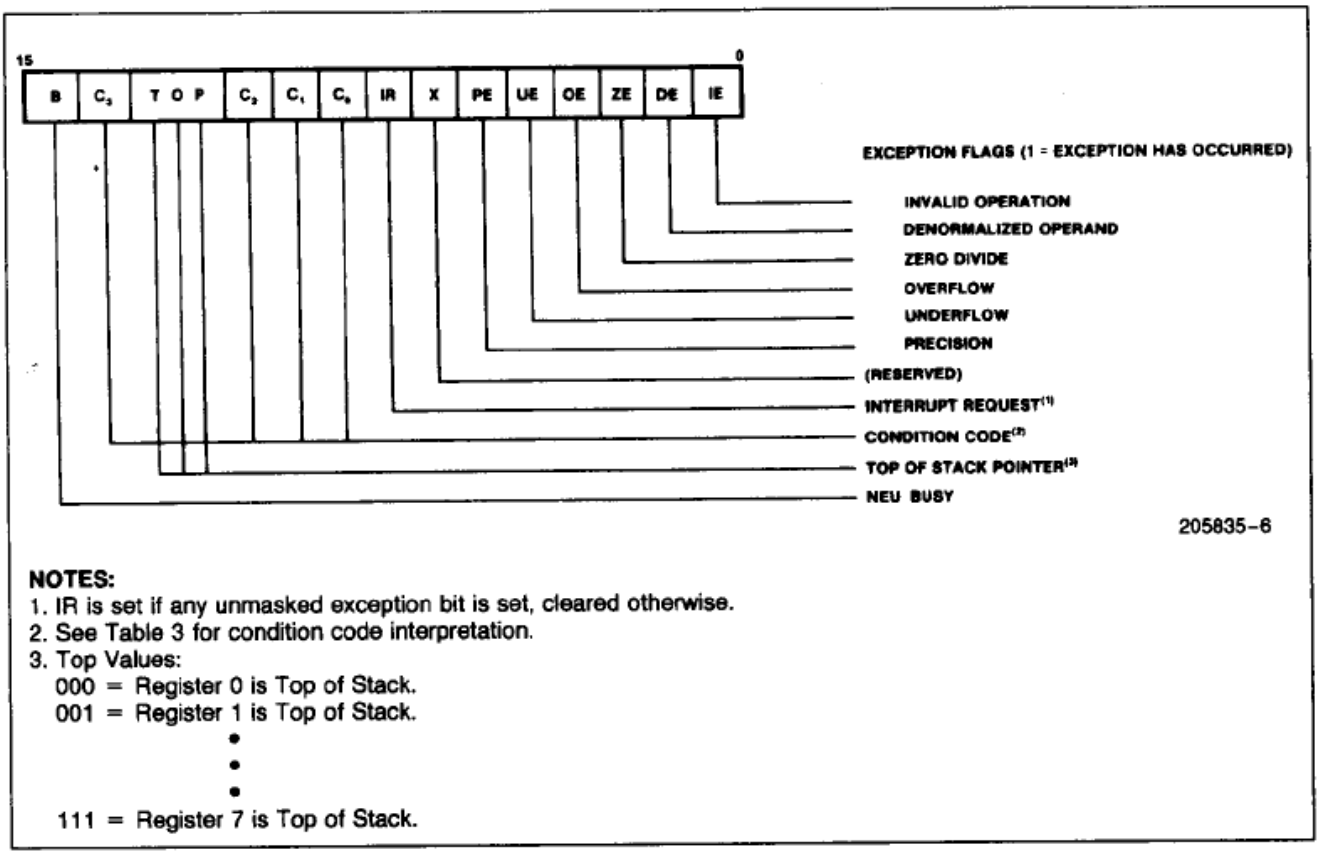


Figure 3. CPU + 8087 Architecture

Los registros internos de la FPU tienen la siguiente diagramación:

Registro de Estado:



NOTES:
1. IR is set if any unmasked exception bit is set, cleared otherwise.
2. See Table 3 for condition code interpretation.
3. Top Values:
   000 = Register 0 is Top of Stack.
   001 = Register 1 is Top of Stack.
   •
   •
   •
   111 = Register 7 is Top of Stack.

Figure 6. 8087 Status Word

Tag Word:

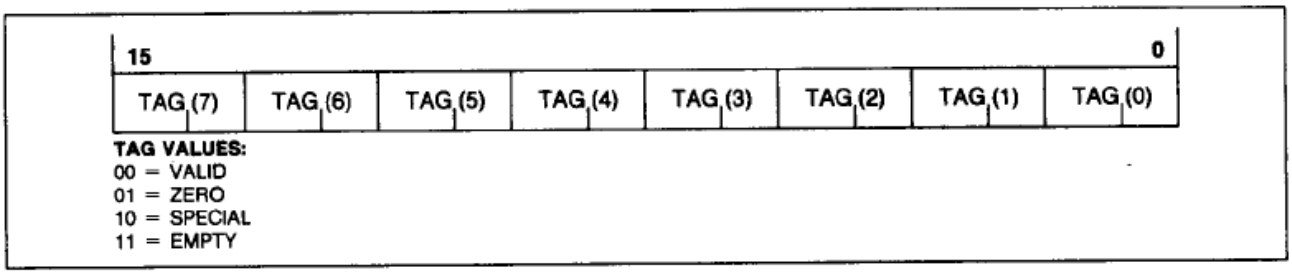


Figure 7. 8087 Tag Word

Registro de Control:

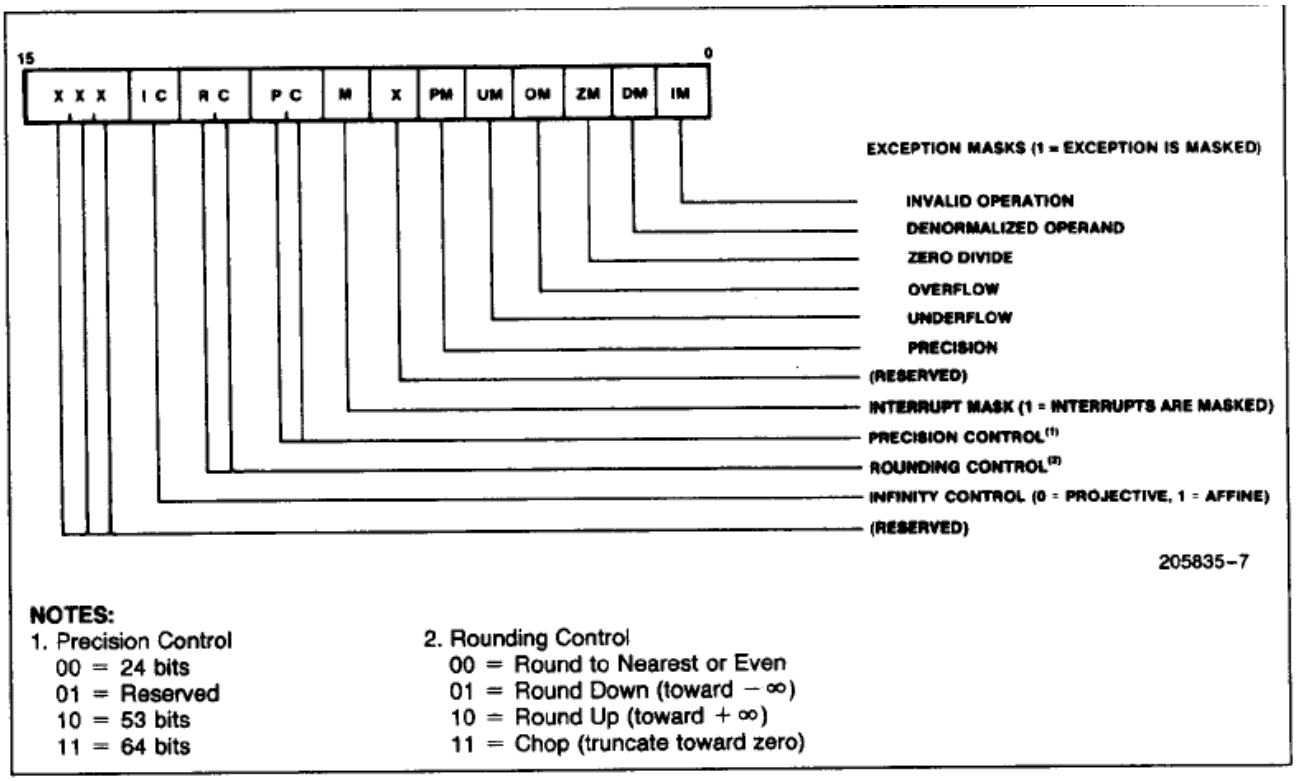


Figure 9. 8087 Control Word

Requerimientos

Como requerimiento un estudio profundo del proyecto actual, para emprender el diseño de su refactorización.
Con el propósito de que el sistema sea multiplataforma se usará código estándar (C y ASM) con compiladores disponbles en las multiples sistemas operativos.
Al ser un software con propósitos educativos, es menester una interfaz gráfica usable y entendible, que aporte al usuario inexperto ayuda contextual sobre las información que se está brindando. Por ejemplo, mensajes emergentes (tool tip text) que expliquen los acrónimos de las distintas banderas de estado y control.
La interfaz entre el middleware en C y el nivel ensamblador se realizará bajo la estructura propuesta por el libro *PC Assembly Language* del Dr. Paul Carter[1].
La interfaz entre el middleware y el lenguaje de alto nivel debe ser transparente y segura, evitando desprolijidades (que representan potenciales agujeros de seguridad) como la interacción a través de archivo de texto plano externo.

Tecnologías a emplear

En concordancia con el estado del proyecto actual, los conocimientos propios y la potencia de las herramientas disponibles se optará por el siguiente conjunto de software:

- Rutinas en lenguaje Ensamblador basado en el compilador NASM v2.05
- Middleware en C compilado con GCC para sistemas Linux y MINGW para sistemas Microsoft Windows.
- Interfaces gráficas basada en la biblioteca GTK 2.0 a través del wrapper PyGTK. Esta biblioteca es nativa en sistemas Linux con entorno Gnome, pero puede ser ejecutado sobre otros gestores de escritorio y sobre sistemas Windows.
- Generación de interfaces gráficas mediante Glade que describe la interfaz de usuario mediante archivos XML
- Python como lenguaje de alto nivel. Es un lenguaje interactivo, que permite *testing on the fly* del código durante el desarrollo. Consta de herramientas avanzadas para la realización de pruebas de depuración mediante la metodología Test Driven Development (desarrollo guiado por pruebas).

1 http://www.drpaulcarter.com/pcasm/

Metodologías

Extreme Programming (XP)

Es un enfoque de la ingeniería de software formulado por Kent Beck. Es una de las metodologías ágiles mas utilizadas. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los cambios de requeisitos sobre la marcha son un aspecto natural e inevitable de la metodología XP y esta contiene una gran facilidad de adaptarse a cambios
Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software.

TDD

Es una práctica de programación que involucra otras dos prácticas: Escribir las pruebas y la refactorización de código. En este caso se eligió realizar pruebas unitarias, con las que se iban mejorando y corrigiendo las distintas funcionalidades del código. Se buscaba el fallo inicial de la prueba y luego se realizaba la refactorización del código a testear hasta que funcionara .

SISTEMA DE CONTROL DE VERSIONES

Todo el desarrollo se realizará sobre un repositorio de control de versiones, gestionado con el software **Subversion**. Al ser un proyecto libre usaremos el servicio Google Code, que integra control de versiones y otras herramientas.

AUTODOCUMENTACIÓN

Todo el código, en sus distintos niveles, será documentado. A través de una herramienta de parseo se autogenerará una documentación exportables a distintos formatos como html (navegable a través de un navegador web), pdf, etc.

Documento de requerimientos

1. Elementos necesarios para el funcionamiento de la FPU:
    - Pila ST 8 registros de 80 bits
    - tags 8 tags de 3 bits, asociados uno a uno con los registros ST de la pila
    - Registro de Estado
    - Registro de Control.
    - FLAGS X86

2. Interfaz Gráfica:
    - Sencilla e integrada.
    - Ayuda contextual de los distintos elementos.
    - Disponer de una entrada donde colocar los comandos
    - Disponer de un botón de deshacer
    - Disponer de botón de salida
    - Disponer de botón de ejecutar los comandos
    - Disponer de botón de reinicio

3. Reiniciar la FPU
    - Disponer de opción de reiniciar la FPU

4. Introducción de comandos
    - estos comandos deben pertenecer a un set de instrucciones de un mínimo 20 instrucciones de FPU a elección
5. Ejecutar Comandos:
    - El usuario debe poder requerir la ejecución de los comandos introducidos

Requerimientos Internos

Se debe poder leer el valor de cabecera de la pila real, mediante la invocación de rutinas de bajo nivel.
Se debe poder convertir entre los diversos tipos de datos binario, decimal, flotante para poder manejar los elementos y mostrarselos al usuario
Se debe constar con algún elemento de conversión entre tipos para su manejo.

Se debe adaptar la entrada de texto de la consola interactiva de entrada para poder realizar la interpretación del código y posterior ejecución.
Se debe realizar la interpretación del código.
Se debe mantener el estado anterior de la FPU para poder realizar la vuelta atrás del

último comando.

Bibliografía

- CHARTE OJEDA, FRANCISCO. - Programación en Ensamblador - 2003
- CARTER, PAUL. - Lenguaje ensamblador para PC – 2004
- Building GUI applications with Python, GTK and Glade - http://video.google.com/videoplay?docid=5838951374743244232
- Documentación de Libglade http://www.jamesh.id.au/software/libglade/
- http://www.gtk.org/documentation.html
- http://www.python.org/doc/
- INTEL, Architecture Software Developer Manual. Volumen 2 Instruction set reference
- INTEL, FPU 8087 math coprocessor
- http://www.rz.uni-karlsruhe.de/rz/docs/VTune/reference/
- http://en.wikipedia.org/wiki/Floating_point_unit
- http://es.wikipedia.org/wiki/Intel_8087
- http://en.wikipedia.org/wiki/Intel_8087