Floating Point Numbers

CSC03B3

UNIVERSITY OF JOHANNESBURG

Outline

Outline

1. IEEE Representation
   - IEEE Representation
   - IEEE Conversion
2. Floating Point Unit
   - Floating Point Unit
   - FPU Instructions
   - Loading items into the FPU
   - Getting items from the FPU
   - FPU Operations
3. FPU Examples
   - Simple Example
   - Full Example

IEEE Representation

UNIVERSITY OF JOHANNESBURG

IEEE Single Precision Floating Point Representation

Floating point numbers have a particular representation in the 80x86 architecture.

- Single precision floating point numbers relate to a **`float`** in **C++** (32-bits)
- Double precision floating point numbers relate to a **`double`** in **C++** (64-bits)

Single precision floating point numbers have the following format:

S	E	E	E	E	E	E	E	E	F	…	F
31	30	29	28	27	26	25	24	23	22	…	0

S	01 bit	Sign bit
E	08 bits	Exponent
F	23 bits	Mantissa

Outline

IEEE Representation

IEEE Representation

IEEE Conversion

Floating Point Unit

Floating Point Unit

FPU Instructions

Loading items into the FPU

Getting items from the FPU

FPU Operations

FPU Examples

Simple Example

Full Example

IEEE Single Precision Floating Point Conversion

Conversion from base 10 to IEEE Single Precision Representation

Convert 78.375 to binary:

2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
1	0	0	1	1	1	0	0	1	1

The number must then be represented in Scientific Notation:

$1\underline{001110}.011_2 = 1.001110011_2 \times 2^6$

Why 6? because we moved the decimal 6 positions!

Now you piece it together based on a few simple rules:

S	0	Positive number 0, negative number 1
E	10000101	Exponent of 2 (6) + 127 (bias value) in binary
F	00111001100000000000000	Digits after decimal point padded with zeros

SEEE EEEE EFFF FFFF FFFF FFFF FFFF FFFF

0100 0010 1001 1100 1100 0000 0000 0000_2 - convert to hexadecimal

4 2 9 C C 0 0 0_{16}

$0x429CC000_{16}$ - 0x is part of the convention used.

Floating Point Unit

Floating Point Unit

- The Floating Point Unit (FPU) on the CPU that allows for floating point calculations. It is independent from the rest of the CPU.
- Floating point number in IEEE representation or integers can be transferred to the FPU.
- If an integer is sent to the FPU it is translated into a floating point number.
- Operations can then be performed on these floating point numbers.

FPU consists of:

- Eight (8) registers
- 80 bits long (32-bit architecture)
- **ST0**, **ST1**, **ST2**, **ST3**, **ST4**, **ST5**, **ST6**, **ST7**

The FPU registers are arranged in a STACK, and they are accessed in the same way that a stack is accessed. When you put something into the FPU, you place it onto a the register stack When you remove something from the FPU, you remove it from the register stack You don’t need to reference the actual FPU registers by name. ST0 is the top of the stack and ST7 is the bottom of the stack.

FPU Instructions

FL* load instructions:

- **FLD** *memory(real)*
- **FILD** *memory(int)*
- **FBLD** *memory(BCD)*
- **FLD ST(num)**
- **FLD1**
- **FLDZ**
- **FLDPI**

FS* store instructions:

- **FSTP** *memory(real)*
- **FST** *memory(real)*
- **FST ST(num)**
- **FIST** *memory(int)*

Miscellaneous

- **FINIT**
- **FADD**
- **FSUB**
- **FMUL**
- **FDIV**
- **FSIN**
- **FCOS**
- **FTAN**

Loading items into the FPU

FL* instructions:

FLD *memory(real)*	Push a real value from memory onto the FPU stack.
FILD *memory(int)*	Push an integer value from memory onto the FPU stack.
FBLD *memory(BCD)*	Push a Binary Coded Decimal (BCD) value from memory onto the FPU stack.
FLD *ST(num)*	Push a value from **St(num)** register onto the stack.
FLD1	Push 1 onto the FPU stack
FLDZ	Push 0 onto the FPU stack
FLDPI	Push π onto the FPU stack

Getting items from the FPU

FS* instructions:

FSTP *memory(real)* Pop the value from the top of the FPU stack.

FST *memory(real)* Copy the value off the top of the FPU stack.

FST *ST(num)* Copy the value from **ST0** and place it in **ST(num)**.

FIST *memory(int)* Copy the value from the top of the stack and convert it into an integer in memory.

Outline

IEEE Representation

IEEE Representation
IEEE Conversion

Floating Point Unit

Floating Point Unit
FPU Instructions
Loading items into the FPU
Getting items from the FPU
FPU Operations

FPU Examples

Simple Example
Full Example

FPU Operations

Any trigonometric functions work with radians!

FINIT Initialize the FPU. Only need to call this once per program

FADD Pop **ST0** and **ST1**, add them together and push the result to the stack

FSUB Pop **ST0** and **ST1**, subtract them and push the result to the stack

FMUL Pop **ST0** and **ST1**, multiply them and push the result to the stack

FDIV Pop **ST0** and **ST1**, divide them and push the result to the stack

FSIN Pop **ST0** and push the sine of the value popped

FCOS Pop **ST0** and push the cosine of the value popped

FTAN Pop **ST0** and push the tangent of the value popped

Many more operations not listed (see textbook, chapter 7 for more)

Outline

IEEE Representation
- IEEE Representation
- IEEE Conversion

Floating Point Unit
- Floating Point Unit
- FPU Instructions
- Loading items into the FPU
- Getting items from the FPU
- FPU Operations

FPU Examples
- Simple Example
- Full Example

FPU Examples

Simple Example

```
.DATA
; REAL4 represents a C-type FLOAT
    value1  REAL4  3.1415
    value2  REAL4  1.0
    result  REAL4  0.0
.CODE
_start:
  ; FINIT initialises the FPU
  finit
  ; Push VALUE1 onto the FPU
  fld  value1
  ; Push VALUE2 onto the FPU
  fld  value2
  fadd
  ; Fetch the result and store in RESULT
  fst  result
  ; Display the result on the screen
  push  result
  call  OutputFloat
```

Outline

IEEE Representation

IEEE Representation

IEEE Conversion

Floating Point Unit

Floating Point Unit

FPU Instructions

Loading items into the FPU

Getting items from the FPU

FPU Operations

FPU Examples

Simple Example

Full Example

Full Example I

```
.386
.MODEL FLAT
INCLUDE io.inc
ExitProcess PROTO NEAR32 stdcall, dwExitCode:DWORD
.STACK  4096
.DATA
    nl      BYTE    10, 0 ; newline for formatting
    fTemp   REAL4   ?      ; floating point variable
.CODE
; formula (9/5 * C) + 32
_convert PROC NEAR32
  ; code on next slide
_convert ENDP

_start:
  ; Create the stack frame
  PUSH  ebp
  MOV    ebp, esp
  ; call convert(27)
  PUSH 27
  CALL _convert
  ; Display eax, eax has integer answer
  PUSH eax
  CALL OutputInt
  ; Newline to separate values
  LEA    ebx, nl
  PUSH  ebx
  CALL  OutputStr
  ; Display ftemp, fTemp has floating point answer
  PUSH  fTemp
  CALL  OutputFloat
  ; Newline to separate values
  LEA    ebx, nl
  PUSH  ebx
  CALL  OutputStr
  ; Destroy the stack frame
  MOV    esp,  ebp
  POP    ebp
  ; Exit
  push    0
  call    ExitProcess
PUBLIC _start
END
```

Outline

IEEE Representation

IEEE Representation

IEEE Conversion

Floating Point Unit

Floating Point Unit

FPU Instructions

Loading items into the FPU

Getting items from the FPU

FPU Operations

FPU Examples

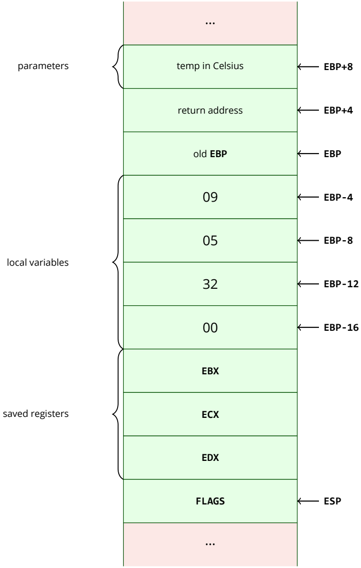
Simple Example

Full Example

Full Example II

```
; int convert(celcius) - formula (9/5 * C) + 32
_convert PROC NEAR32
  ; Entry code
  PUSH  ebp
  MOV    ebp, esp
  SUB    esp, 16    ; 4 Local DWORDS
  PUSH  ebx
  PUSH  ecx
  PUSH  edx
  PUSHFD

  ; Parameters
  ; [ebp+ 8] - celcius -  4 bytes
  ; Local variables
  ; No names needed as we just using them for the conversion
  MOV    [ebp- 4], DWORD PTR 9
  MOV    [ebp- 8], DWORD PTR 5
  MOV    [ebp-12], DWORD PTR 32
  MOV    [ebp-16], DWORD PTR 0    ; to save integer answer

  ; Initialise floating point unit
  FINIT

  ; Load values onto FPU in the correct order
  ; Order is important as the FPU oprates with a stack
  FILD  DWORD PTR [ebp-12]  ; 32
  FILD  DWORD PTR [ebp+ 8]  ; C
  FILD  DWORD PTR [ebp- 4]  ; 9
  FILD  DWORD PTR [ebp- 8]  ; 5
  ; Calculation - keep track of values in comments to make easier
  FDIV  ; 9/5
  FMUL  ; (9/5) * C
  FADD  ; (9/5) * C + 32
  ; Save answers
  FIST  DWORD PTR [ebp-16]  ; get integer answer back
  FSTP  fTemp                ; save floating point answer in global
  MOV    eax, [ebp-16]        ; eax has integer answer

  ; Exit code
  POPFD
  POP    edx
  POP    ecx
  POP    ebx
  MOV    esp, ebp
  POP    ebp
  RET    4          ; params are 4 bytes
_convert ENDP
```

Full Example III


...
parameters
temp in Celsius
EBP+8
return address
EBP+4
old EBP
EBP
local variables
09
EBP-4
05
EBP-8
32
EBP-12
00
EBP-16
saved registers
EBX
ECX
EDX
FLAGS
ESP
...