Computer Structure and Language

Hamid Sarbazi-Azad
Department of Computer Engineering
Sharif University of Technology (SUT)
Tehran, Iran

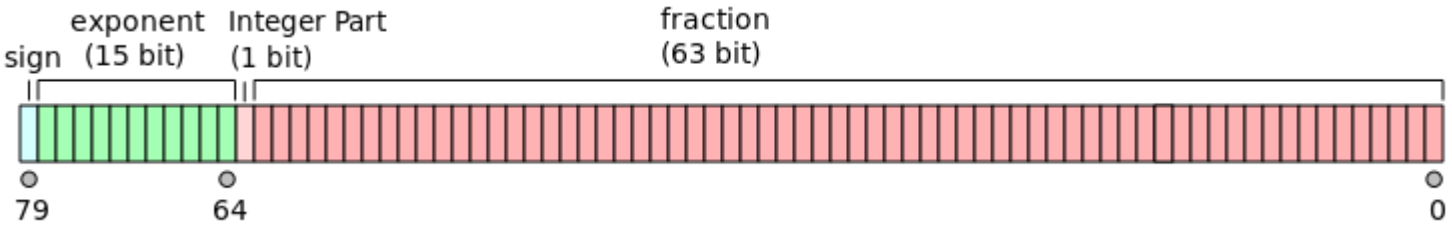
Agenda

- FPU
- Scalar operations
- SIMD operations

FPU

FPU

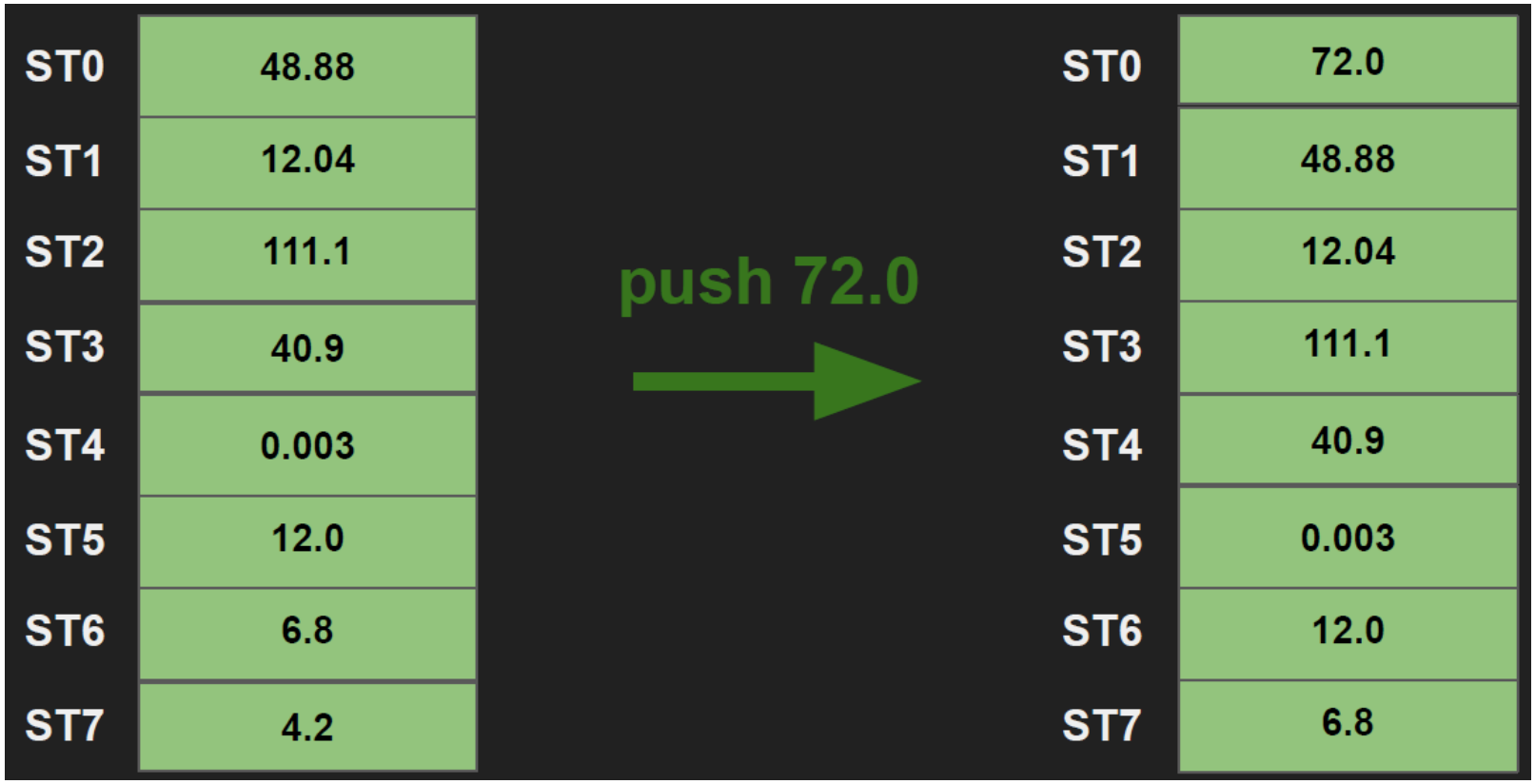
- In early x86 processors, there were no instructions for floating point arithmatics
- x87 coprocessor (Floating Point Unit, FPU) was created to address this issue.
- Later FPU got integrated into main CPUs
- FPU uses extended precision floating point representation (in registers)



https://en.wikipedia.org/wiki/Extended_precision

FPU - Registers

- FPU has 8 registers named st0 to st7
- FPU register file is a stack, st0 is always the top
- FPU operations also affect flags



FPU - Load

Instruction	Effect
FLD mem32 (/mem64/mem80)	Push st0 <- mem
FLD STi	Push ST0 <- STi
FILD mem16 (/mem32/mem64)	Push ST0 <- int2float(mem)
FLD1	Push ST0 <- 1.0
FLDZ	Push ST0 <- 0.0
FLDPI	Push ST0 <- π (the pi number)
FLDL2T/FLDL2E FLDLG2/FLDLN2	

Example:

```
FLD dword [l1]
FLD qword [l2]
```

FPU – Store

Instruction	Effect
FST mem32 (/mem64/mem80)	mem <- st0
FST STi	STi <- ST0
FIST mem16 (/mem32/mem64)	mem <- float2int(ST0)
FSTP dest FISTP dest	similar to FSTP and FISTP but also pops top of stack

Example:

```
FST dword [l1]
FST qword [l2]
```

FPU – Exchange

Instruction	Effect
FXCH STi	ST0 <-> STi

FPU – Arithmatic

Instruction	Effect
FADD src	ST0 += src
FSUB src	ST0 -= src
FMUL src	ST0 *= src
FDIV src	ST0 /= src
FSUBR src	ST0 = src – ST0
FDIVR src	ST0 = src / ST0
	src: STi/mem32/mem64

FPU – Arithmetic

Instruction	Effect
FADDP STi	STi += ST0
FSUBP STi	STi -= ST0
FMULP STi	STi *= ST0
FDIVP STi	STi /= ST0
FSUBRP STi	STi = ST0 – STi
FDIVRP STi	STi = ST0 / Sti
	And pop out top of the stack

FPU – Arithmatic

Instruction	Effect
FIADD src	ST0 += int2float(src)
FISUB src	ST0 -= int2float(src)
FIMUL src	ST0 *= int2float(src)
FIDIV src	ST0 /= int2float(src)
FISUBR src	ST0 = int2float(src) – ST0
FIDIVR src	ST0 = int2float(src) / ST0
	src: mem32/mem64

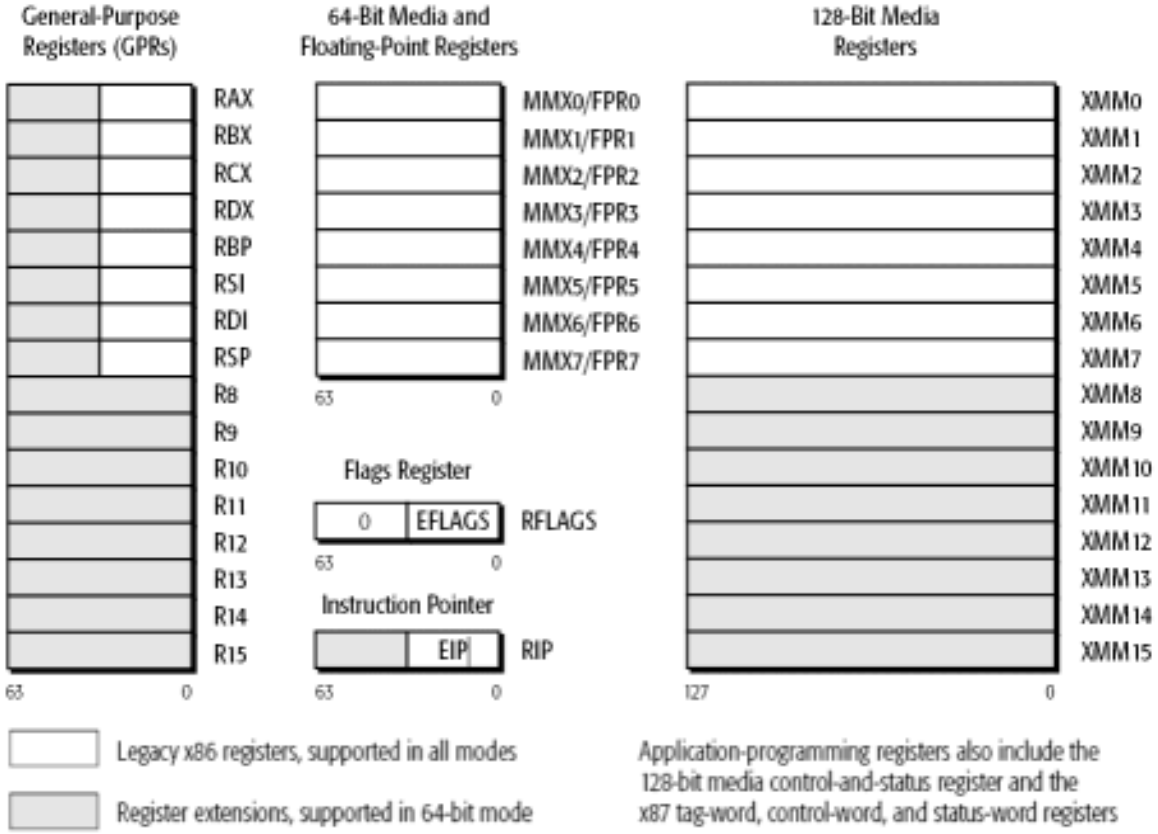
FPU – Comparrison

Instruction	Effect
FCOM src	Compare ST0, src src: mem32/mem64
FCOMP src	Like FCOM Also pops top of the stack
FCOMPP	Compare ST0, ST1 Pop twice
FICOM src	Compare ST0, int2float(src) src: mem32/mem64
FTST	Compare ST0, 0

FPU - Problems

- stack-based floating point computation
- inefficient machine code for complex arithmetic
- needless swaps with the top of stack
- 1999: Intel introduced SSE instruction in Pentium III
  - 128 bit registers XMM0, XMM1, …, XMM7
  - Also capable of doing multiple operations at once
- Currently FPU and SSE coexists and have same physical registers

Registers (x86_64)



https://pvs-studio.com/en/blog/posts/a0029/

FPU – Rest of Instructions

- https://www.felixcloutier.com/x86/
- https://en.wikipedia.org/wiki/X86_instruction_listings#x87_floating-point_instructions

END OF SLIDES