PSI3441 – Arquitetura de Sistemas Embarcados

- Instruções de Desvio – Loop
- Ponto Flutuante
- Multiplicação e Divisão
- Pseudo-Instruções
- Processadores ARM

Escola Politécnica da Universidade de São Paulo

Prof. Gustavo Rehder – grehder@lme.usp.br

Prof. Sergio Takeo – kofuji@usp.br

Prof. Antonio Seabra – acseabra@lsi.usp.br

2017

Socrative

Quiz

Sala: PSI3441

PSI3441 – Arquitetura de Sistemas Embarcados

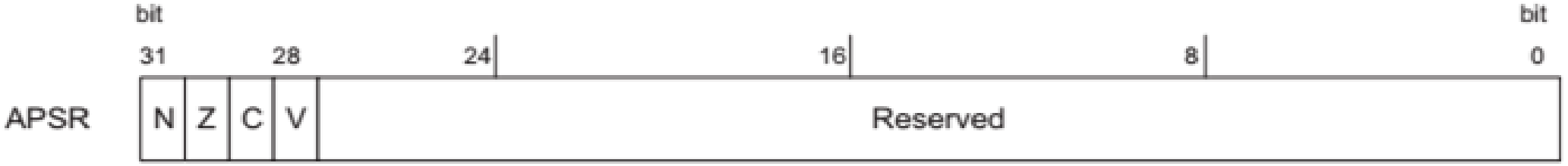
- Instruções de Desvio – Loop

Escola Politécnica da Universidade de São Paulo

2017

Registrador APSR - Flags

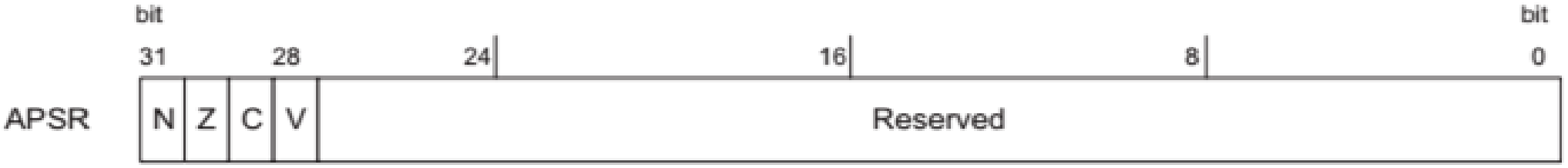
- N (Bit 31) – Flag setado → resultado Negativo
- Z (Bit 30) – Flag setado → resultado Zero ou de mesmo valor em uma comparação
- C (Bit 29) – Flag setado → "Carry" (vai 1) do resultado de soma
- V (Bit 28) – Flag setado → oVerflow do resultado de soma ou subtração



Desvio Condicional - Instruções

B{cond} <Rótulo>

- BNE <Rótulo> ; Branch if Not Equal - Desvio se Z = 0
- BEQ <Rótulo> ; Branch if EQual - Desvio se Z = 1
- BLS <Rótulo> ; Branch if Less or Same - Desvio se Z = 1 ou C = 0
- BHI <Rótulo> ; Branch if HIgher- Desvio se Z = 0 e C = 1
- BCS/BHS <Rótulo> ; Branch if Carry Set/Higher or Same - Desvio se C = 1
- BCC/BLO <Rótulo> ; Branch if Carry Clear/LOwer- Desvio se C = 0



Loop *For* – BNE (Branch if Not Equal)

```
For (i=0, i<10, i++){
...
}
```

```
        MOV     R0, #10          ; carrega o contador
VOLTA   instrução 1
        instrução 2
        ...
        SUBS R0, R0,#1           ; decrementa 1 do contador e checa se o
                                 ; contador chegou a zero, se sim, seta o flag Z.
        BNE VOLTA                ; desvia ao rótulo se flag Z=0.
```

Desvio Condicional com Comparação

Comparação de Números *Unsigned*

```
CMP    Rn, Op2        ; compara os operando e muda os flags. Não tem
                      ; registrador de destino, não muda os operandos.
                      ; operação de subtração (Rn – Op2) para setar os flags
```

Instruction	C	Z
Rn > Op2	1	0
Rn = Op2	1	1
Rn < Op2	0	0

Instruction		Action
BCS/BHS	branch if carry set/branch if higher or same	Branch if Rn ≥ Op2
BCC/BLO	branch if carry clear/branch lower	Branch if Rn < Op2
BEQ	branch if equal	Branch if Rn = Op2
BNE	branch if not equal	Branch if Rn ≠ Op2
BLS	branch if less or same	Branch if Rn ≤ Op2
BHI	branch if higher	Branch if Rn > Op2

Loop *If* – BCC (Branch if Carry Clear)

R1=0x35F;

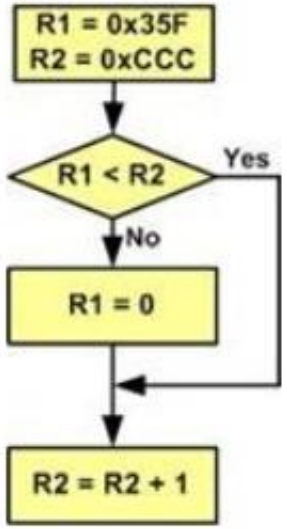
R2=0xCCC;

If (R1>=R2){

R1=0;

}

R2=R2+1



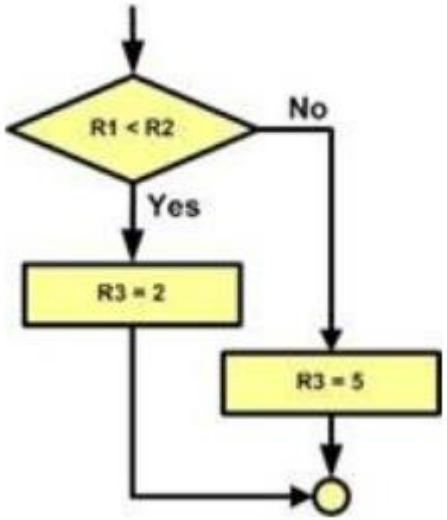
Comparação CMP

Instruction	C	Z
Rn > Op2	1	0
Rn = Op2	1	1
Rn < Op2	0	0

```
            LDR     R1,=35F
            LDR     R2,=CCC
            CMP     R1,R2          ; R1<R2 → C =0
            BCC     SE_SIM         ; desvia para rótulo se flag C=0.
SE_NAO      MOV     R1,#0
SE_SIM      ADD R2, R2, #1
```

Loop *If Else* – BHS (Branch if Higher or Same) e B (Branch)

```
if(R1 < R2)
{
R3 = 2;
}
else
{
R3 = 5;
}
```

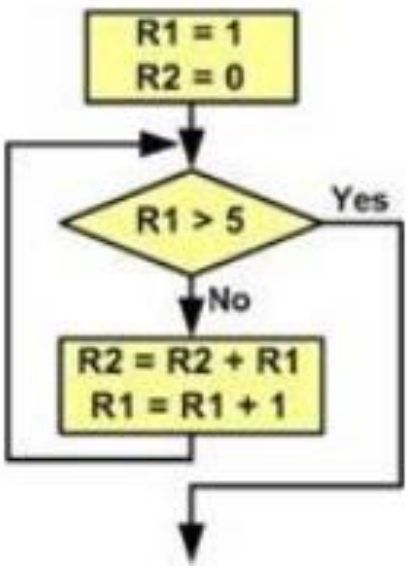


Comparação CMP

Instruction	C	Z
Rn > Op2	1	0
Rn = Op2	1	1
Rn < Op2	0	0

```
        CMP     R1,R2
        BHS     L1          ; desvio se C=1.
        MOV     R3,#2
        B       FIM
L1      MOV     R3,#5
FIM
```

Loop *While* – BHI (Branch if Higher) e B (Branch)




```
int R1 = 1;
int R2 = 0;
while (R1 < 5) {
R2 = R2 + R1;
R1 = R1 + 1;
}
...
```

Comparação CMP

Instruction	C	Z
Rn > Op2	1	0
Rn = Op2	1	1
Rn < Op2	0	0

```
        MOV R1,#1
        MOV R2,#0
L1      CMP R1,#5
        BHI L2          ; desvia para rótulo se C=1 e Z = 0.
        ADD R2,R2,R1
        ADD R1,R1,#1
        B L1
L2      ...
```

Exercício

- Escreva em C o código abaixo:

```
gcd
        CMP     r0, r1
        BEQ     end
        BLT     less
        SUBS    r0, r0, r1
        B       gcd
less
        SUBS    r1, r1, r0
        B       gcd
end
```

O que faz esse código?

BEQ <Rótulo> ; Branch if EQual - Desvio se Z = 1
BLT <Rótulo> ; Branch if Less Than - Desvio se N ≠V

CMP Rn, Op2

Instruction	C	Z
Rn > Op2	1	0
Rn = Op2	1	1
Rn < Op2	0	0

Socrative

Quiz

Sala: PSI3441

PSI3441 – Arquitetura de Sistemas Embarcados

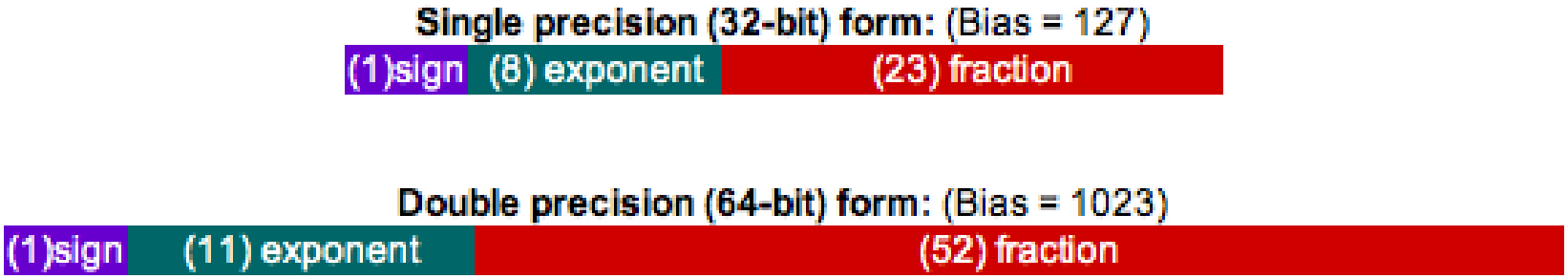
- Ponto Flutuante

Escola Politécnica da Universidade de São Paulo

2017

Representação de Ponto Flutuante (IEEE 754)

- Representa números entre $\pm 1.4 \times 10^{-45}$ a $\pm 3.4 \times 10^{38}$
- Erro de arredondamento
  - 45.45 $\rightarrow$ 45.450000000000284... (double)
    $\rightarrow$ 45.4500007629... (single)

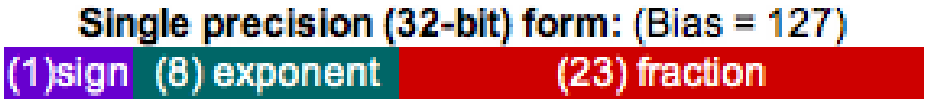


$$(-1)^{\text{sign bit}}(1+\text{fraction}) \times 2^{\text{exponent}} - \text{bias}$$

Conversor: https://www.h-schmidt.net/FloatConverter/IEEE754.html

Representação de Ponto Flutuante (IEEE 754)

- Sinal:
  - Bit 31 = 0 $\rightarrow$ $(-1)^0 = 1$ (Positivo)
  - Bit 31 = 1 $\rightarrow$ $(-1)^1 = -1$ (Negativo)



$$(-1)^{\text{sign bit}}(1+\text{fraction})\times 2^{\text{exponent}} - \text{bias}$$

Conversão de Decimal para Ponto Flutuante (IEEE 754)

- 45.45 (decimal)
  - Converter inteiro para binário: 45 → 10 1101
  - Converter a fração para binário:

0.45 x 2 = **0**.9
0.90 x 2 = **1**.8
0.80 x 2 = **1**.6
0.60 x 2 = **1**.2
0.20 x 2 = **0**.4
0.40 x 2 = **0**.8
0.80 x 2 = **1**.6

Valores se repetem

011100...

45.45 → 101101.011100...
Normalização → $1.01101011100... \times 2^5$

$$(-1)^{\text{sign bit}}(1+\text{fraction})\times 2^{\text{exponent}-\text{bias}}$$

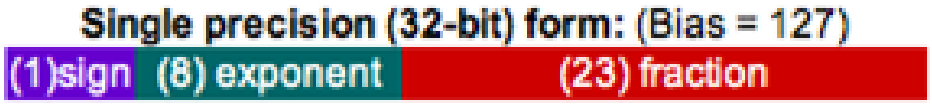
$$(-1)^{0}(1.011100...)\times 2^{132-127} = 1.011100\times 2^{5}$$

0 10000100 01101011100110011001100

Single precision (32-bit) form: (Bias = 127)

(1)sign	(8) exponent	(23) fraction

Conversão de Ponto Flutuante para Decimal



0 10000100 01101011100110011001100

- Sinal: 0 → Positivo
- Expoente: 10000100 → 132
- Fração: 01101011100110011001100

binário decimal

$0.1 = 2^{-1} = 0.5$

$0.01 = 2^{-2} = 0.25$

$0.001 = 2^{-3} = 0.125$

⋮

$$(-1)^{\text{sign bit}}(1+\text{fraction})\times 2^{\text{exponent}-\text{bias}}$$

$$(-1)^{0}(1.\ 011010\ldots)\times 2^{132-127} = 1.\ 01101011100110011001100 \times 2^{5}$$

101101.01110011001100

45 $\quad \mathbf{0}\cdot 2^{-1}+\mathbf{1}\cdot 2^{-2}+\mathbf{1}\cdot 2^{-3}+\mathbf{1}\cdot 2^{-4}+\mathbf{0}\cdot 2^{-5}+\ldots = 0.25+0.125+0.0625+\ldots$

45.4500007629...

Soma ou Subtração com Ponto Flutuante

- Checar se existem zeros
- Igualar expoentes
- Soma ou subtração
- Normalizar resultados

- Exemplo (8 bits – bias = 7)
  - X+Y
  - X = 7 = 0 1001 110 $\rightarrow$ 1.110×2^2
  - Y = 1 = 0 0111 000 $\rightarrow$ $1.000 \times 2^0 \rightarrow 0.01000 \times 2^2$

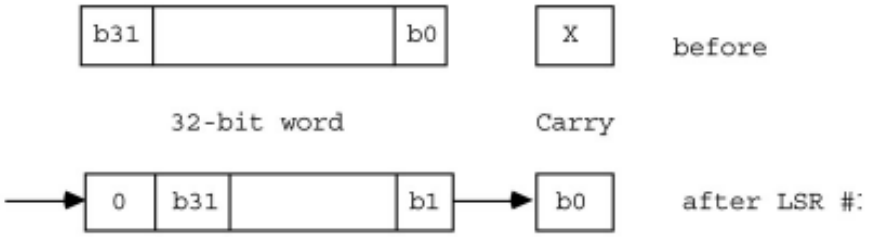
Mesmo Expoente

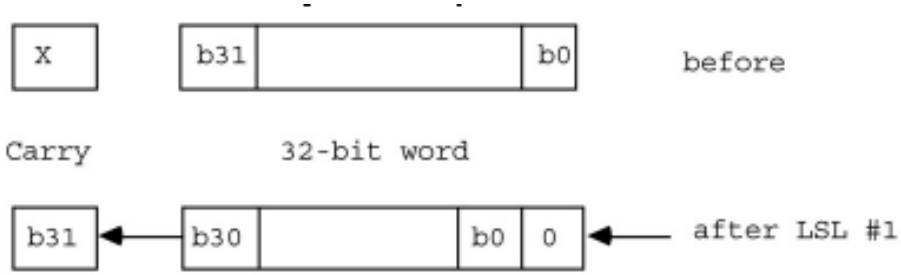
$$\begin{array}{r} 1.110 \times 2^2 \\ +\ 0.010 \times 2^2 \\ \hline 10.000 \times 2^2 \end{array} \rightarrow 1.000 \times 2^3 \rightarrow 0\ 1010\ 000$$

$X + Y = 8$

Como se iguala os expoentes?

- Logical Shift Right (LSR) ou Logical Shift Leght (LSL)






```
LSR     Rd, Rm, Op2

LDR     R2, =0x0001 0000
LSR     R0, R2, #8

R0 = 0000 0010
C = 0
```

```
LSL     Rd, Rm, Op2

LDR     R2, =0x0001 0000
LSL     R0, R2, #8

R0 = 0100 0000
C = 0
```

Addição - inteiros e ponto flutuante

- Comparação

```
int a = 2;
int b = 2;
int c = 0;
c = a+b;
```

22 ticks de clock
com o KL25Z

```
float a = 2.0;
float b = 2.0;
float c = 0.0;
c = a+b;
```

98 ticks de clock
com o KL25Z

PSI3441 – Arquitetura de Sistemas Embarcados

- Multiplicação e Divisão

Escola Politécnica da Universidade de São Paulo

2017

Multiplicação (MUL) – Unsigned Int

```
MUL   Rd, Rn, Opt2   ; Rd = Rn x Opt2
```

- Cortex M0+ possui duas implementações possíveis:
  - Rápida – executa em um único ciclo → Usado pelo KL25Z (Hardware)
  - Pequena – multiplicador iterativo executa em 32 ciclos

    → Instruções add/sub/shift usadas quando não existe a opção de 1-ciclo (Software)

- A Instrução MUL realiza a multiplicação de dois números de 32 bits (Rn e Opt2) e retorna em Rd um número de 32 bits (LSBs).
- Multiply and Accumulated (MLA) → Instrução não disponível no Cortex M0+

```
MLA   Rd, Rm, Rs, Rn        ; Rd = Rm x Rs +Rn
```

Multiplicação Long

- Instruções que multiplicam números de 32 bits e retornam números de 64 bits

UMULL – Unsigned Multiply Long
UMLAL – Unsigned Multiply with Accumlate Long
SMULL – Signed Multiply Long
SMLAL – Signed Multiply with Accumlate Long

Disponíveis a partir do Cortex M3

Multiplicação- inteiros e ponto flutuante

(Cortex M0+)

- Comparação

```
int a = 2;
int b = 2;
int c = 0;
c = a x b;
```

22 ticks de clock
com o KL25Z

Utiliza MUL
(Hardware)

```
float a = 2.0;
float b = 2.0;
float c = 0.0;
c = a x b;
```

178 ticks de clock
com o KL25Z

Utiliza add/sub/shift
(Software)

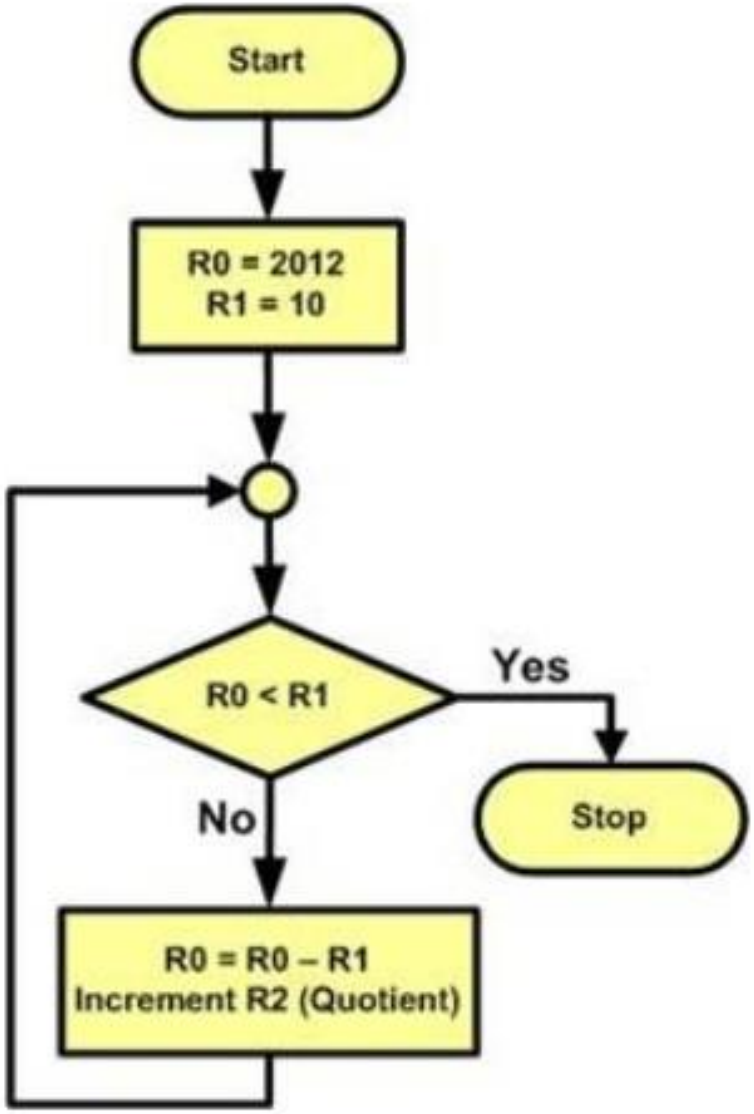
Divisão – Signed and Unsigned Int

- Signed Divide: SDIV{cond} Rd, Rn, Rm
- Unsigned Divide: UDIV{cond} Rd, Rn, Rm

Disponíveis a partir do Cortex M3 (Hardware)

Como fazer divisão no Cortex M0+ ?

Divisão Inteiro – Cortex M0+ (Software)



Exemplo: R2 = R0/R1

```
int R0 = 2012;
int R1 = 10;
int R2 = 0;

while (R0 >= R1)
{
R0 = R0 - R1;
R2 = R2 + 1;
}
```

Quando R0 for maior ou igual a R1, R2 = 201 e R0 = 2

Divisão - inteiros e ponto flutuante (Cortex M0+)

- Comparação

```
int a = 2;
int b = 2;
int c = 0;
c = a / b;
```

88 ticks de clock com o KL25Z

Software

```
float a = 2.0;
float b = 2.0;
float c = 0.0;
c = a / b;
```

429 ticks de clock com o KL25Z

Software

PSI3441 – Arquitetura de Sistemas Embarcados

- Pseudo-Instruções

Escola Politécnica da Universidade de São Paulo

2017

Assembly Pseudo-Instruções

- Auxiliam na programação
- Não utilizam memória
- São indicações para o Assembler traduzir o código de Assembly para Código de Máquina

Directive	Description
AREA	Instructs the assembler to assemble a new code or data section
END	Informs the assembler that it has reached the end of a source file.
ENTRY	Declares an entry point to a program.
EQU	Gives a symbolic name to a numeric constant, a register-relative value or a PC-relative value.
INCLUDE	It adds the contents of a file to our program.

PSI3441 – Arquitetura de Sistemas Embarcados

- Processadores ARM

Escola Politécnica da Universidade de São Paulo

2017

Instruction Sets - ARM

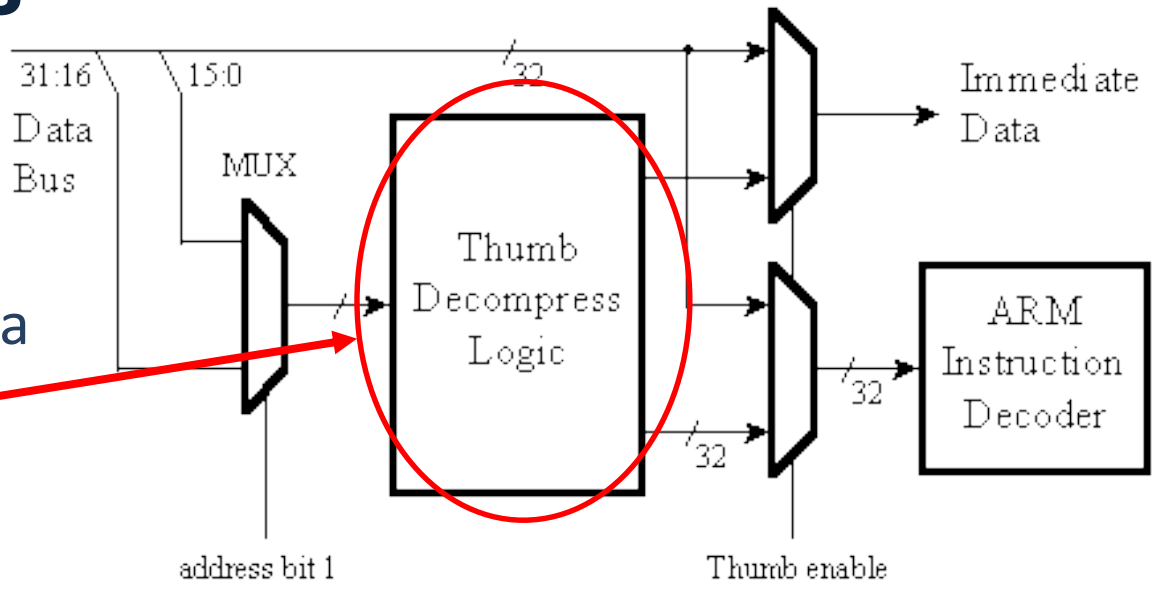
- Processadores ARM possuem diversos grupos de instruções dependendo da versão do processador:
  - 32-bit ARM instruction set
  - 16-bit *Thumb* instruction set
  - 16/32-bit *Thumb-2* instruction set

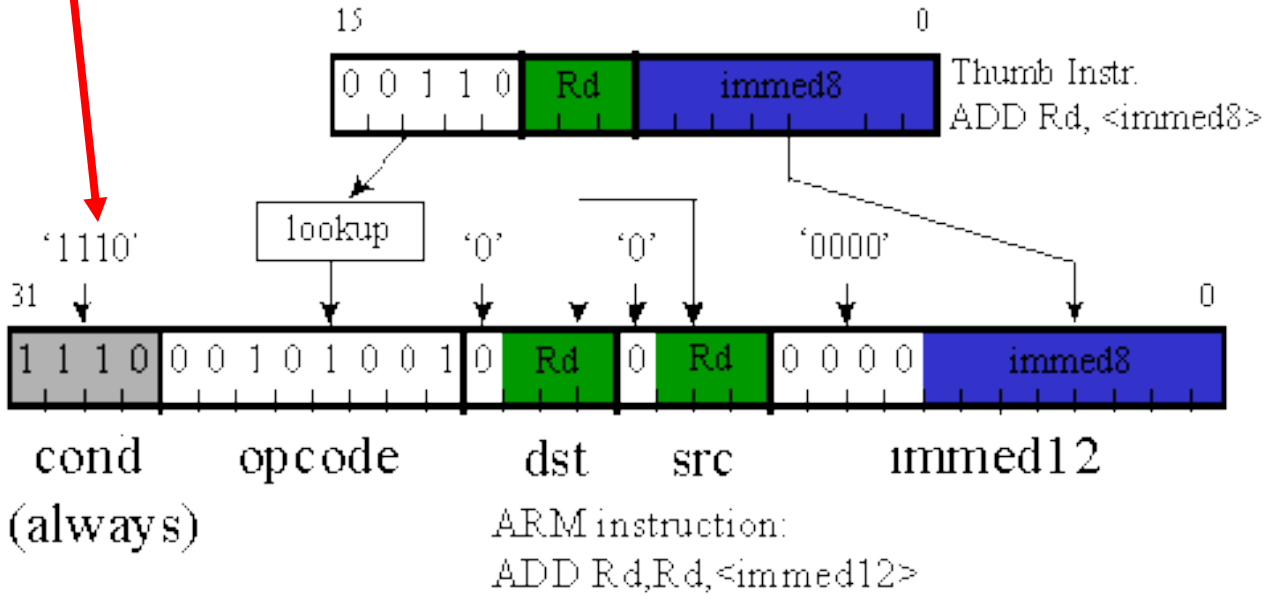
  } Cortex M0+

  - *Jazelle DBX* for Java byte codes (multi-tasking Java Virtual Machine)
  - *NEON* 64/128-bit SIMD instruction set (Media ARM Cortex A e R52)
  - *VFP* vector floating point instruction set (FPU)

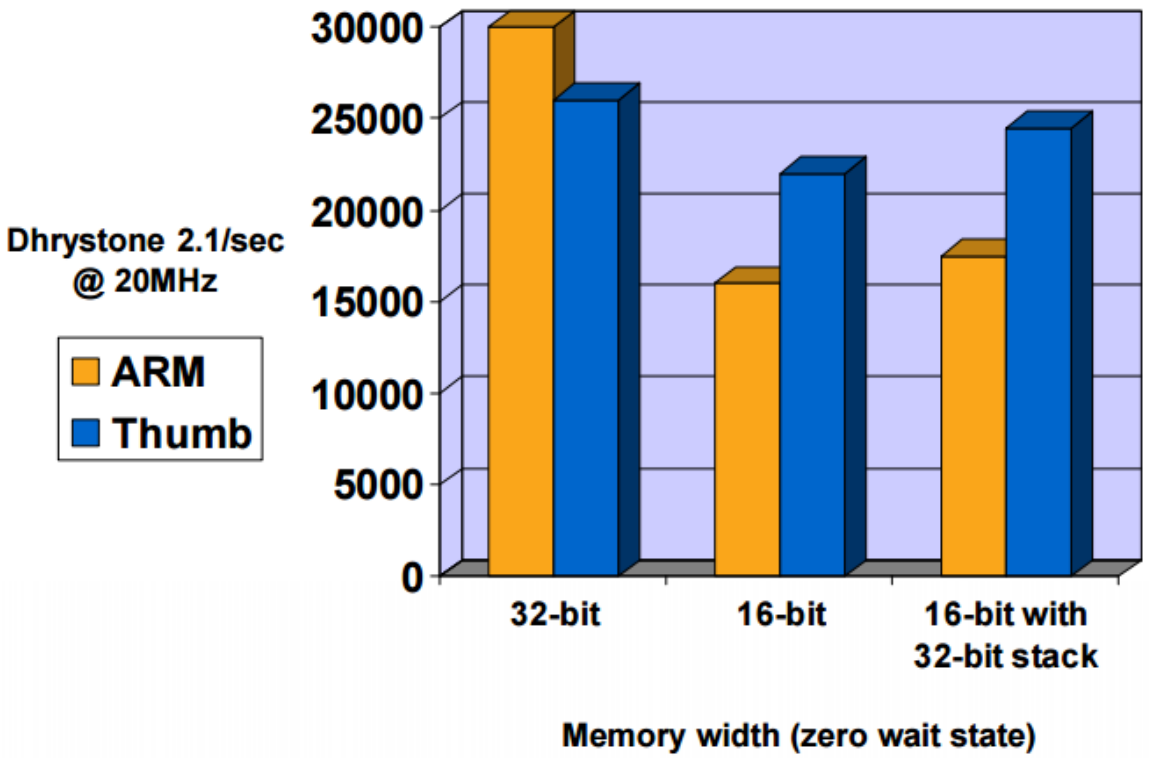
Instruction Sets

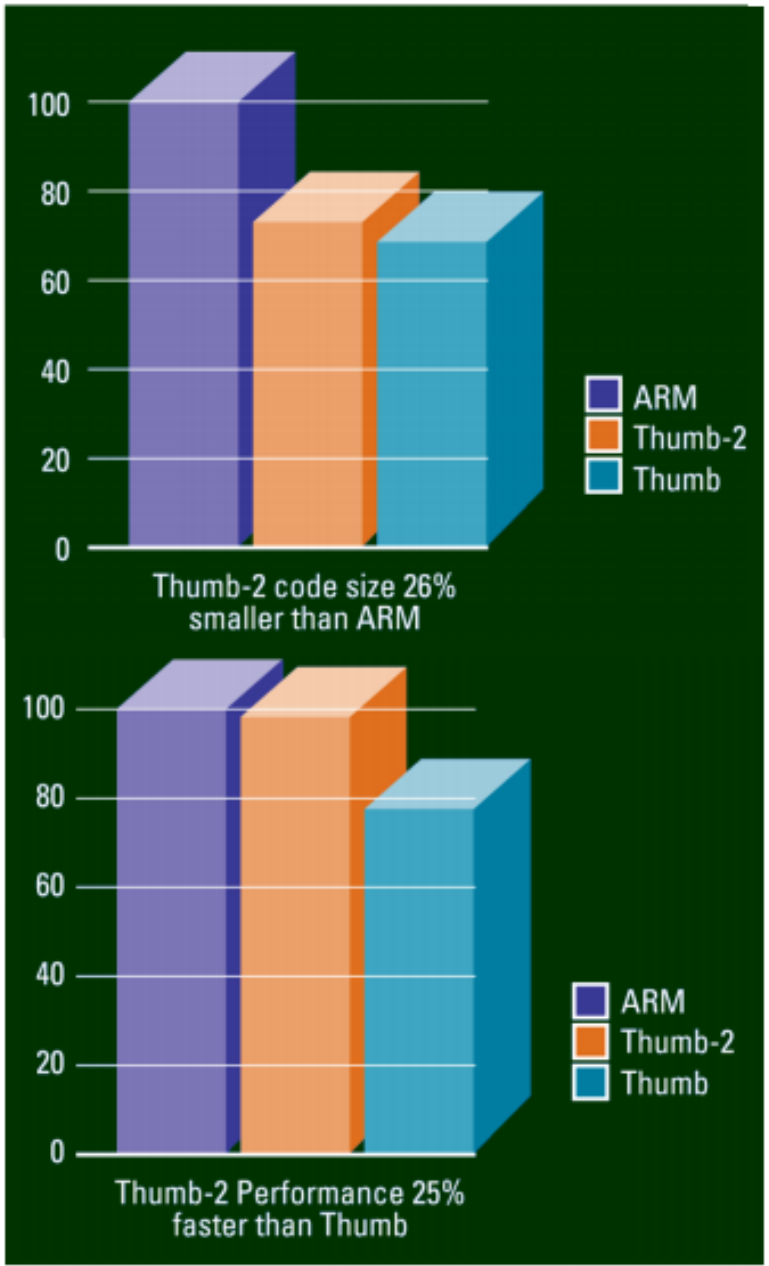
- Thumb – instrução precisa ser descomprimida
- Todas as instruções ARM podem ser condicionais





Comparação entre Instruction Sets





Família ARM Cortex-M

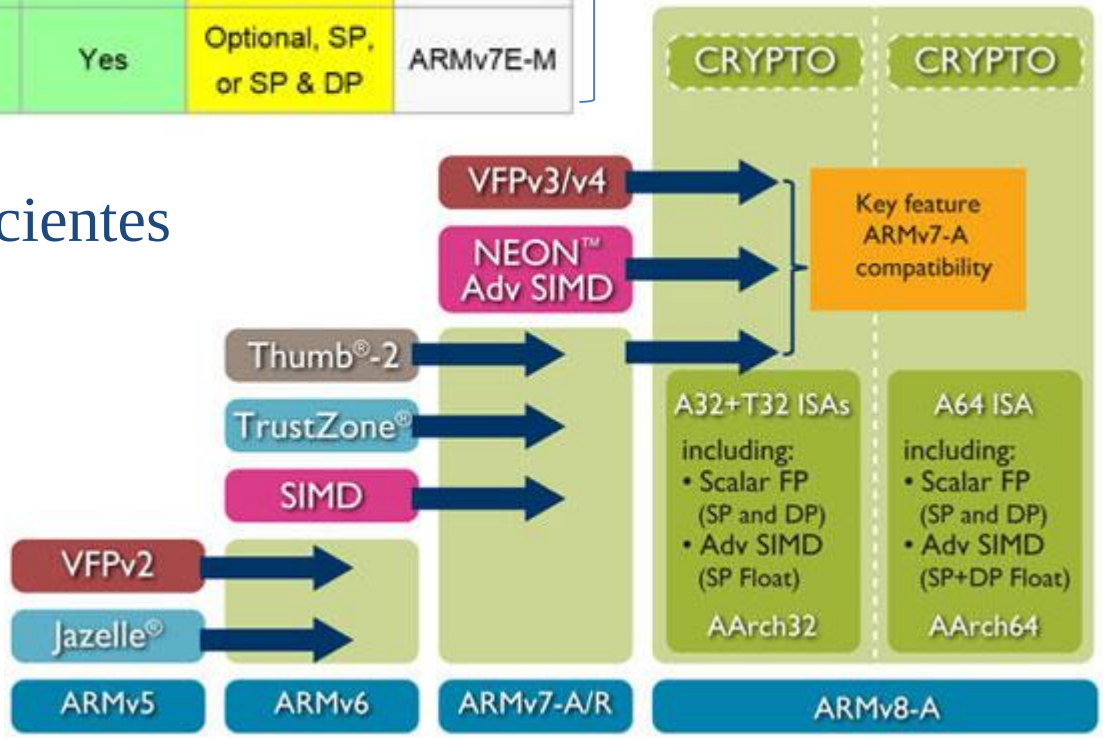
ARM Cortex-M instruction sets[6][7]

ARM Cortex-M	Thumb	Thumb-2	Hardware multiply	Hardware divide	Saturated math	DSP extensions	Floating-Point Unit (FPU)	ARM architecture
Cortex-M0[1]	Entire	Subset	1 or 32 cycle	No	No	No	No	ARMv6-M
Cortex-M0+[2]	Entire	Subset	1 or 32 cycle	No	No	No	No	ARMv6-M
Cortex-M1[3]	Entire	Subset	3 or 33 cycle	No	No	No	No	ARMv6-M
Cortex-M3[4]	Entire	Entire	1 cycle	Yes	Yes	No	No	ARMv7-M
Cortex-M4[5]	Entire	Entire	1 cycle	Yes	Yes	Yes	Optional, SP	ARMv7E-M
Cortex-M7	Entire	Entire	1 cycle	Yes	Yes	Yes	Optional, SP, or SP & DP	ARMv7E-M

Von Neumann

Harvard

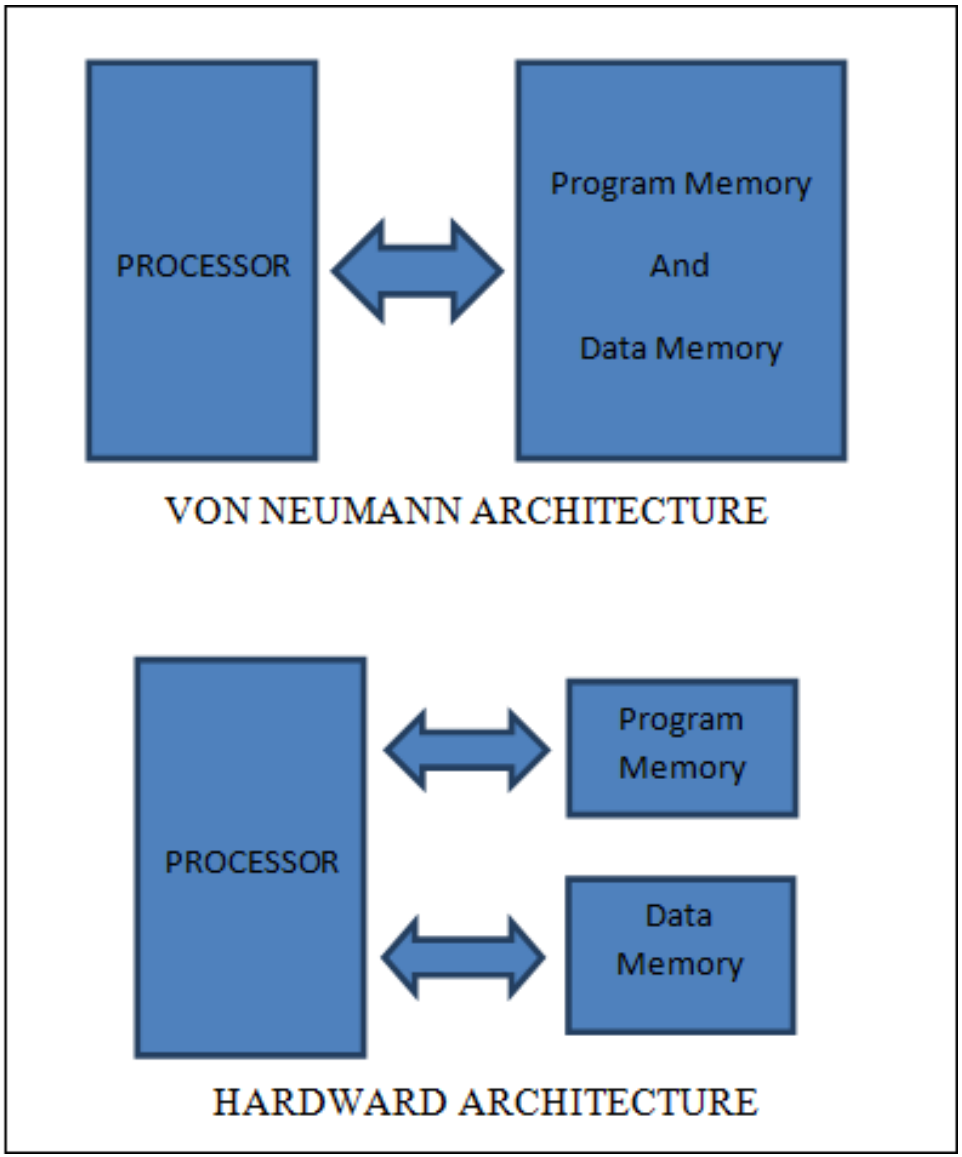
Saturated Math → Algoritmos eficientes para processamento de sinais



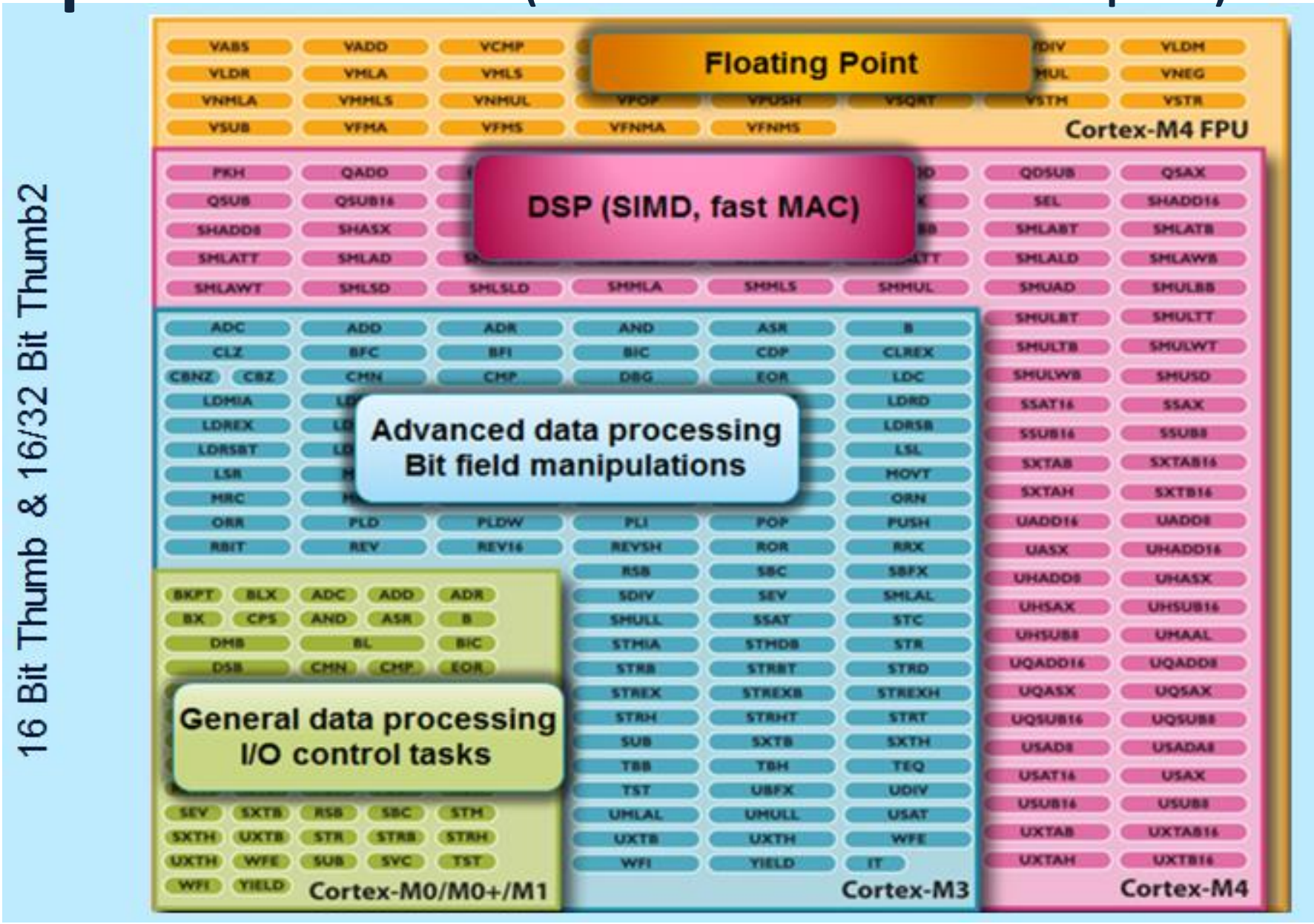
Von Neumann vs Harvard

- 1 vs 2 Barramentos
  - Custo
  - Complexidade
  - Acesso simultâneo a instrução e dados
  - Implementação em microcontraladores
  - ~~Velocidade~~

Cache e Pipeline



Arquitetura RISC (Reduced Instruction Set Computer)



Instruções x86 - CISC (Reduced Instruction Set Computer)

AAA,AAD,AAM,AAS,ADC,ADCX,ADD,ADDPD,ADDPS,ADDSD,ADDSS,ADDSUBPD,ADDSUBPS,ADOX,AESDEC,AESDECLAST,AESENC,AESENCLAST,AESIMC,AES
KEYGENASSIST,AND,ANDN,ANDNPD,ANDNPS,ANDPD,ANDPS,ARPL,BEXTR,BLENDPD,BLENDPS,BLENDVPD,BLENDVPS,BLSI,BLSMSK,BLSR,BOUND,BSF,BSR,
BSWAP,BT,BTC,BTR,BTS,BZHI,CALL,CBW,CDQ,CDQE,CLAC,CLC,CLD,CLFLUSH,CLI,CLTS,CMC,CMOVcc,CMP,CMPPD,CMPPS,CMPS,CMPSB,CMPSD,CMPSD,CMP
SQ,CMPSS,CMPSW,CMPXCHG,CMPXCHG16B,CMPXCHG8B,COMISD,COMISS,CPUID,CQO,CRC32,CVTDQ2PD,CVTDQ2PS,CVTPD2DQ,CVTPD2PI,CVTPD2PS,CVTPI2
PD,CVTPI2PS,CVTPS2DQ,CVTPS2PD,CVTPS2PI,CVTSD2SI,CVTSD2SS,CVTSI2SD,CVTSI2SS,CVTSS2SD,CVTSS2SI,CVTTPD2DQ,CVTTPD2PI,CVTTPS2DQ,CVTTPS2PI,
CVTTSD2SI,CVTTSS2SI,CWD,CWDE,DAA,DAS,DEC,DIV,DIVPD,DIVPS,DIVSD,DIVSS,DPPD,DPPS,EMMS,ENTER,EXTRACTPS,F2XM1,FABS,FADD,FADDP,FBLD,FBS
TP,FCHS,FCLEX,FCMOVcc,FCOM,FCOMI,FCOMIP,FCOMP,FCOMPP,FCOS,FDECSTP,FDIV,FDIVP,FDIVR,FDIVRP,FFREE,FIADD,FICOM,FICOMP,FIDIV,FIDIVR,FILD,
FIMUL,FINCSTP,FINIT,FIST,FISTP,FISTTP,FISUB,FISUBR,FLD,FLD1,FLDCW,FLDENV,FLDL2E,FLDL2T,FLDLG2,FLDLN2,FLDPI,FLDZ,FMUL,FMULP,FNCLEX,FNINIT
,FNOP,FNSAVE,FNSTCW,FNSTENV,FNSTSW,FPATAN,FPREM,FPREM1,FPTAN,FRNDINT,FRSTOR,FSAVE,FSCALE,FSIN,FSINCOS,FSQRT,FST,FSTCW,FSTENV,FSTP,
FSTSW,FSUB,FSUBP,FSUBR,FSUBRP,FTST,FUCOM,FUCOMI,FUCOMIP,FUCOMP,FUCOMPP,FWAIT,FXAM,FXCH,FXRSTOR,FXSAVE,FXTRACT,FYL2X,FYL2XP1,HA
DDPD,HADDPS,HLT,HSUBPD,HSUBPS,IDIV,IMUL,IN,INC,INS,INSB,INSD,INSERTPS,INSW,INT3,INT,INTO,INVD,INVLPG,INVPCID,IRET,IRETD,JMP,Jcc,LAHF,LAR,L
DDQU,LDMXCSR,LDS,LEA,LEAVE,LES,LFENCE,LFS,LGDT,LGS,LIDT,LLDT,LMSW,LOCK,LODS,LODSB,LODSD,LODSQ,LODSW,LOOP,LOOPcc,LSL,LSS,LTR,LZCNT
,MASKMOVDQU,MASKMOVQ,MAXPD,MAXPS,MAXSD,MAXSS,MFENCE,MINPD,MINPS,MINSD,MINSS,MONITOR,MOV,MOV,MOV,MOVAPD,MOVAPS,MOVBE,M
OVD,MOVDDUP,MOVDQ2Q,MOVDQA,MOVDQU,MOVHLPS,MOVHPD,MOVHPS,MOVLHPS,MOVLPD,MOVLPS,MOVMSKPD,MOVMSKPS,MOVNTDQ,MOVNTDQA,
MOVNTI,MOVNTPD,MOVNTPS,MOVNTQ,MOVQ,MOVQ,MOVQ2DQ,MOVS,MOVSB,MOVSD,MOVSD,MOVSHDUP,MOVSLDUP,MOVSQ,MOVSS,MOVSW,MOVSX,M
OVSXD,MOVUPD,MOVUPS,MOVZX,MPSADBW,MUL,MULPD,MULPS,MULSD,MULSS,MWAIT,MULX,MWAIT,NEG,NOP,NOT,OR,ORPD,ORPS,OUT,OUTS,OUTSB,O
UTSD,OUTSW,PABSB,PABSD,PABSW,PACKSSDW,PACKSSWB,PACKUSDW,PACKUSWB,PADDB,PADDD,PADDQ,PADDSB,PADDSW,PADDUSB,PADDUSW,PADDW
,PALIGNR,PAND,PANDN,PAUSE,PAVGB,PAVGW,PBLENDVB,PBLENDW,PCLMULQDQ,PCMPEQB,PCMPEQD,PCMPEQQ,PCMPEQW,PCMPESTRI,PCMPESTRM,PCM
PGTB,PCMPGTD,PCMPGTQ,PCMPGTW,PCMPISTRI,PCMPISTRM,PDEP,PEXT,PEXTRB,PEXTRD,PEXTRQ,PEXTRW,PHADDD,PHADDSW,PHADDW,PHMINPOSUW,PH
SUBD,PHSUBSW,PHSUBW,PINSRB,PINSRD,PINSRQ,PINSRW,PMADDUBSW,PMADDWD,PMAXSB,PMAXSD,PMAXSW,PMAXUB,PMAXUD,PMAXUW,PMINSB,PMIN
SD,PMINSW,PMINUB,PMINUD,PMINUW,PMOVMSKB,PMOVSX,PMOVZX,PMULDQ,PMULHRSW,PMULHUW,PMULHW,PMULLD,PMULLW,PMULUDQ,POP,POPA,P
OPAD,POPCNT,POPF,POPFD,POPFQ,POR,PREFETCHW,PREFETCHWT1,PREFETCHh,PSADBW,PSHUFB,PSHUFD,PSHUFHW,PSHUFLW,PSHUFW,PSIGNB,PSIGND,PSI
GNW,PSLLD,PSLLDQ,PSLLQ,PSLLW,PSRAD,PSRAW,PSRLD,PSRLDQ,PSRLQ,PSRLW,PSUBB,PSUBD,PSUBQ,PSUBSB,PSUBSW,PSUBUSB,PSUBUSW,PSUBW,PTEST,
PUNPCKHBW,PUNPCKHDQ,PUNPCKHQDQ,PUNPCKHWD,PUNPCKLBW,PUNPCKLDQ,PUNPCKLQDQ,PUNPCKLWD,PUSH,PUSHA,PUSHAD,PUSHF,PUSHFD,PXOR,
RCL,RCPPS,RCPSS,RCR,RDFSBASE,RDGSBASE,RDMSR,RDPMC,RDRAND,RDSEED,RDTSC,RDTSCP,REP,REPE,REPNE,REPNZ,REPZ,RET,ROL,ROR,RORX,ROUNDP
D,ROUNDPS,ROUNDSD,ROUNDSS,RSM,RSQRTPS,RSQRTSS,SAHF,SAL,SAR,SARX,SBB,SCAS,SCASB,SCASD,SCASW,SETcc,SFENCE,SGDT,SHL,SHLD,SHLX,SHR,S
HRD,SHRX,SHUFPD,SHUFPS,SIDT,SLDT,SMSW,SQRTPD,SQRTPS,SQRTSD,SQRTSS,STAC,STC,STD,STI,STMXCSR,STOS,STOSB,STOSD,STOSQ,STOSW,STR,SUB,SU
BPD,SUBPS,SUBSD,SUBSS,SWAPGS,SYSCALL,SYSENTER,SYSEXIT,SYSRET,TEST,TZCNT,UCOMISD,UCOMISS,UD2,UNPCKHPD,UNPCKHPS,UNPCKLPD,UNPCKLP
S,VBROADCAST,VCVTPH2PS,VCVTPS2PH,VERR,VERW,VEXTRACTF128,VEXTRACTI128,VFMADD132PD,VFMADD132PS,VFMADD132SD,VFMADD132SS,VFMAD
D213PD,VFMADD213PS,VFMADD213SD,VFMADD213SS,VFMADD231PD,VFMADD231PS,VFMADD231SD,VFMADD231SS,VFMADDSUB132PD,VFMADDSUB132PS,V
FMADDSUB213PD,VFMADDSUB213PS,VFMADDSUB231PD,VFMADDSUB231PS,VFMSUB132PD,VFMSUB132PS,VFMSUB132SD,VFMSUB132SS,VFMSUB213PD,VFM
SUB213PS,VFMSUB213SD,VFMSUB213SS,VFMSUB231PD,VFMSUB231PS,VFMSUB231SD,VFMSUB231SS,VFMSUBADD132PD,VFMSUBADD132PS,VFMSUBADD213P
D,VFMSUBADD213PS,VFMSUBADD231PD,VFMSUBADD231PS,VFNMADD132PD,VFNMADD132PS,VFNMADD132SD,VFNMADD132SS,VFNMADD213PD,VFNMADD
213PS,VFNMADD213SD,VFNMADD213SS,VFNMADD231PD,VFNMADD231PS,VFNMADD231SD,VFNMADD231SS,VFNMSUB132PD,VFNMSUB132PS,VFNMSUB132S
D,VFNMSUB132SS,VFNMSUB213PD,VFNMSUB213PS,VFNMSUB213SD,VFNMSUB213SS,VFNMSUB231PD,VFNMSUB231PS,VFNMSUB231SD,VFNMSUB231SS,VGAT
HERDPD,VGATHERDPS,VGATHERQPD,VGATHERQPS,VINSERTF128,VINSERTI128,VMASKMOV,VPBLENDD,VPBROADCAST,VPERM2F128,VPERM2I128,VPERMD,
VPERMILPD,VPERMILPS,VPERMPD,VPERMPS,VPERMQ,VPGATHERDD,VPGATHERDQ,VPGATHERQD,VPGATHERQQ,VPMASKMOV,VPSLLVD,VPSLLVQ,VPSRA
VD,VPSRLVD,VPSRLVQ,VTESTPD,VTESTPS,VZEROALL,VZEROUPPER,WAIT,WBINVD,WRFSBASE,WRGSBASE,WRMSR,XABORT,XACQUIRE,XADD,XBEGIN,XC
HG,XEND,XGETBV,XLAT,XLATB,XOR,XORPD,XORPS,XRELEASE,XRSTOR,XRSTORS,XSAVE,XSAVEC,XSAVEOPT,XSAVES,XSETBV,XTEST,

663 Instruções

RISK vs CISC

- RISK (Load-Store)
  - Instruções e endereçamento simples
  - Instruções executadas em um ciclo de clock
  - Tamanho de instrução fixo
  - Pipeline
  - Complexidade
- CISC (Register-Memory)
  - Endereçamento complexo
  - Rico em instruções
  - Instruções executadas em mais de um ciclo de clock
  - Tamanho da instrução de tamanho variável
  - Sem pipeline