Professor
Edson Pedro FERLIN

Arquitetura de Computador

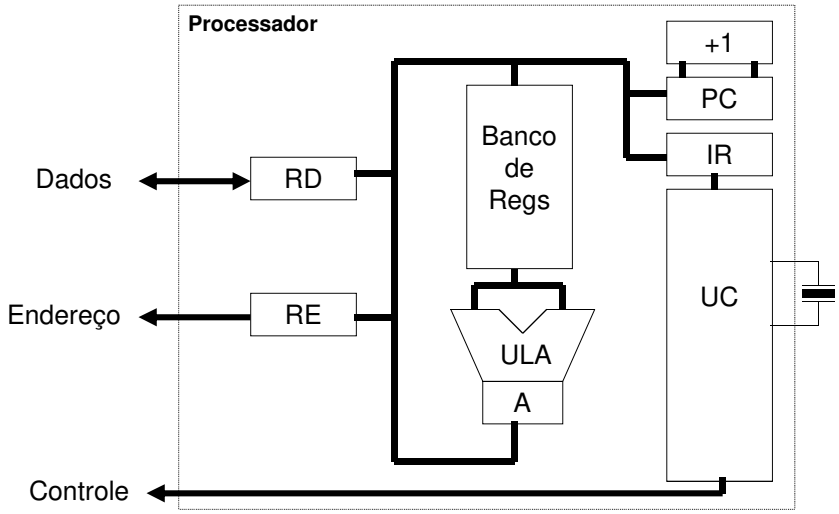
Microarquitetura

Prof. Edson Pedro Ferlin

Professor
Edson Pedro FERLIN

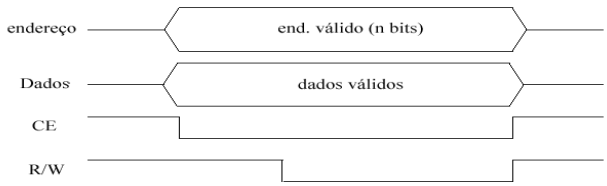
Arquitetura de Computador

- Objetivos
  - Apresentar os conceitos de Microprogramação
- Conteúdos
  - Definições
  - Microarquitetura
  - Microprogramação
  - Macroarquitetura

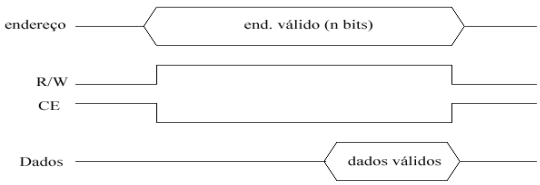


Ciclos de Leitura e Escrita da Memória

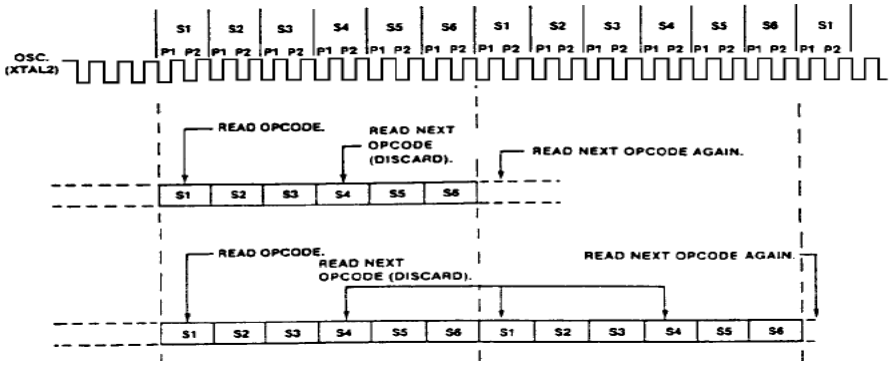
Escrita



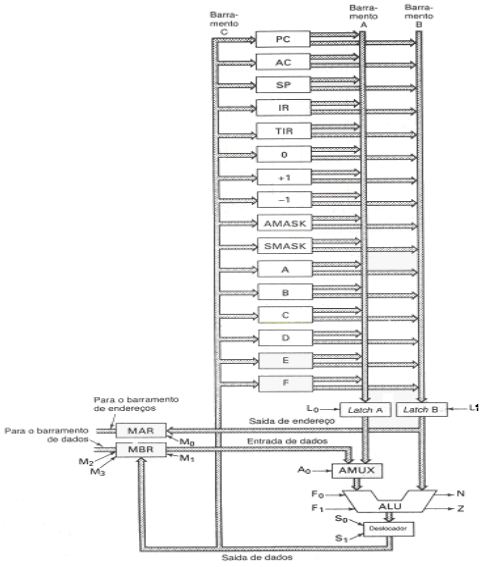
Leitura



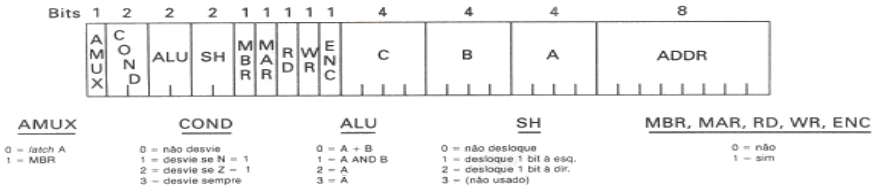
Ciclos de Máquina e de *Clock*



Fluxo de Dados



Palavra de Controle - Microinstrução



AMUX
0 = latch A
1 = MBR

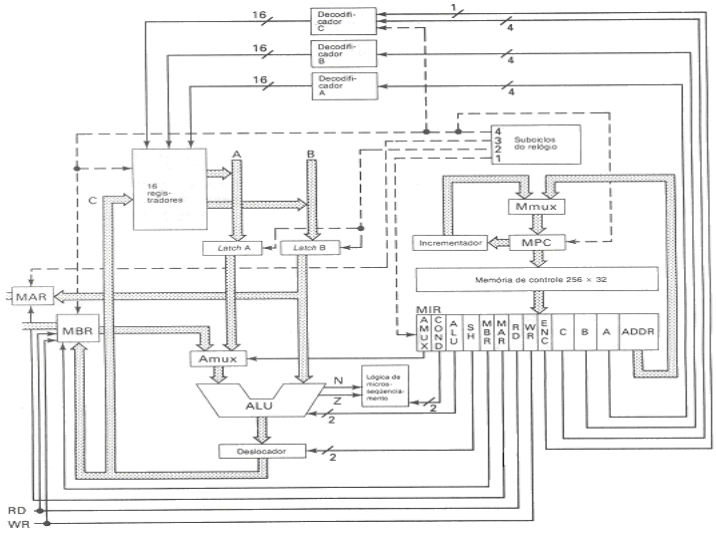
COND
0 = não desvie
1 = desvie se N = 1
2 = desvie se Z = 1
3 = desvie sempre

ALU
0 = A + B
1 = A AND B
2 = A
3 = A

SH
0 = não desloque
1 = desloque 1 bit à esq.
2 = desloque 1 bit à dir.
3 = (não usado)

MBR, MAR, RD, WR, ENC
0 = não
1 = sim

Microarquitetura



Conjunto de Instruções

Binário	Mnemônico	Instrução	Significado
0000xxxxxxxxxxxx	LODD	Carrega direto	*ac* := *m* [*x*]
0001xxxxxxxxxxxx	STOD	Armazena direto	*m* [*x*] := *ac*
0010xxxxxxxxxxxx	ADDD	Adiciona direto	*ac* := *ac* + *m* [*x*]
0011xxxxxxxxxxxx	SUBD	Subtrai direto	*ac* := *ac* − *m* [*x*]
0100xxxxxxxxxxxx	JPOS	Desvia se positivo	**if** *ac* ≥ 0 **then** *pc* := *x*
0101xxxxxxxxxxxx	JZER	Desvia se zero	**if** *ac* = 0 **then** *pc* := *x*
0110xxxxxxxxxxxx	JUMP	Desvia	*pc* := *x*
0111xxxxxxxxxxxx	LOCO	Carega constante	*ac* := *x* (0 ≤ *x* ≤ 4095)
1000xxxxxxxxxxxx	LODL	Carga total	*ac* := *m* [*sp* + *x*]
1001xxxxxxxxxxxx	STOL	Armazena local	*m* [*x* + *sp*] := *ac*
1010xxxxxxxxxxxx	ADDL	Adiciona local	*ac* := *ac* + *m* [*sp* + *x*]
1011xxxxxxxxxxxx	SUBL	Subtrai local	*ac* := *ac* − *m* [*sp* + *x*]
1100xxxxxxxxxxxx	JNEG	Desvia se negativo	**if** *ac* < 0 **then** *pc* := *x*
1101xxxxxxxxxxxx	JNZE	Desvia se não zero	**if** *ac* ≠ 0 **then** *pc* := *x*
1110xxxxxxxxxxxx	CALL	Chama procedimento	*sp* := *sp* − 1; *m* [*sp*] := *pc*; *pc* := *x*
1111000000000000	PSHI	Empilha indireto	*sp* := *sp* − 1; *m* [*sp*] := *m* [*ac*]
1111001000000000	POPI	Desempilha indireto	*m* [*ac*] := *m* [*sp*]; *sp* := *sp* + 1
1111010000000000	PUSH	Coloca na pilha	*sp* := *sp* + 1; *m* [*sp*] := *ac*
1111011000000000	POP	Retira da pilha	*ac* := *m* [*sp*]; *sp* := *sp* + 1
1111100000000000	RETN	Retorna	*pc* := *m* [*sp*]; *sp* := *sp* + 1
1111101000000000	SWAP	Troca ac, sp	*tmp* := *ac*; *ac* := *sp*; *sp* := *tmp*
11111100yyyyyyyy	INSP	Incrementa sp	*sp* := *sp* + *y* (0 ≤ *y* ≤ 255)
11111110yyyyyyyy	DESP	Decrementa sp	*sp* := *sp* − *y* (0 ≤ *y* ≤ 255)

xxxxxxxxxxxx é um endereço de máquina de 12 bits; na coluna 4, ele é chamado de *x*.
yyyyyyyy é uma constante de 8 bits; na coluna 4, ele é chamado de *y*.

Microprograma

```
0: mar := pc; rd;                          {loop principal}
1: pc := pc+ 1; rd;                        {incremente pc}
2: ir := mbr; if n then goto 28;           {salva, decodifica mbr}
3: tir := lshift (ir + ir); if n then goto 19;
4: tir := lshift (tir); if n then goto 11; {000x ou 001x?}
5: alu := tir; if n then goto 9;           {0000 ou 0001?}

6: mar := ir; rd;                          {0000 = LODD}
7: rd;
8: ac := mbr; goto 0;

9: mar := ir; mbr := ac; wr;               {0001 = STOD}
10: wr ;goto 0;

11: alu := tir; if n then goto 15;         {0010 ou 0011?}

12: mar := ir; rd;                         {0010 = ADDD}
13: rd;
14: ac := mbr + ac; goto 0;

15: mar := ir; rd;                         {0011 = SUBD}
16: ac := ac + 1; rd;                      {Nota: x − y = x + 1 + not y}
17: a:= inv (mbr);
18: ac := ac + a; goto 0;

19: tir := lshift (tir); if n then goto 25; {010x ou 011x?}
20: alu := tir; if n then goto 23;         {0100 ou 0101?}

21: alu := ac; if n then goto 0;           {0100 = JPOS}
22: pc := band (ir, amask); goto 0;        {faça o desvio}

23: alu := ac; if z then goto 22;          {0101 = JZER}
24: goto 0;                                {desvio falhou}

25: alu:= tir; if n then goto 27;          {0110 ou 0111?}
26: pc := band (ir, amask); goto 0;        {0110 = JUMP}

27: ac := band (ir, amask); goto 0;        {0111 = LOCO}

28: tir := lshift (ir + ir); if n then goto 40; {10xx ou 11xx?}
29: tir := lshift (tir); if n then goto 35;     {100× ou 101x?}
30: alu := tir; if n then goto 33;              {1000 ou 1001?}

31: a := ir + sp;                          {1000 = LODL}
32: mar := a; rd; goto 7;

33: a := ir + sp;                          {1001 = STOL}
34: mar := a; mbr := ac; wr; goto 10;

35: alu := tir; if n then goto 38;         {1010 ou 1011?}

36: a := ir + sp;                          {1010 = ADDL}
37: mar := a; rd; goto 13;

38: a := ir + sp;                          {1011 = SUBL}
39: mar := a; rd; goto16;
```

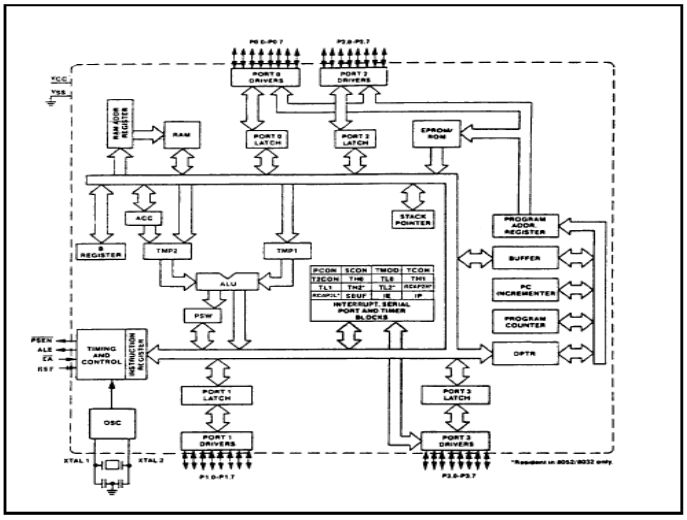
```
40: tir := lshift (tir); if n then goto 46;   {110x ou 111x?}
41: alu := tir; if n then goto 44;            {1100 ou 1101?}

42: alu := ac; if n then goto 22;             {1100 = JNEG}
43: goto 0;

44: alu := ac; if z then goto 0;              {1101 = JNZE}
45: pc := band (ir, amask); then goto 0;

46: tir := lshift (tir); if n then goto 50;

47: sp := sp + (−1);                          {1110 = CALL}
48: mar := sp; mbr := pc; wr;
49: pc := band (ir, amask); wr; goto 0;

50: tir := lshift (tir); if n then goto 65;   {1111, examine endereço}
51: tir := lshift (tir); if n then goto 59;
52: alu := tir; if n then goto 56;

53: mar := ac; rd;                            {1111000 = PSHI}
54: sp := sp + (−1); rd;
55: mar := sp; wr; goto 10;

56: mar := sp; sp := sp + 1; rd;              {1111001 = POPI}
57: rd;
58: mar := ac; wr; goto 10;

59: alu := tir; if n then goto 62;

60: sp := sp + (−1);                          {1111010 = PUSH}
61: mar := sp; mbr := ac; wr; goto 10;

62: mar := sp; sp := sp + 1; rd;              {1111011 = POP}
63: rd;
64: ac:= mbr; goto 0;

65: tir := lshift (tir); if n then goto 73;
66: alu := tir; if n then goto 70;

67: mar := sp; sp := sp + 1; rd;              {1111100 = RETN}
68: rd;
69: pc:= mbr; goto 0;

70: a := ac;                                  {1111101 = SWAP}
71: ac := sp;
72: sp := a; goto 0;

73: alu := tir; if n then goto 76;

74: a := band (ir, smask);                    {1111110 = INSP}
75: sp := sp + a; goto 0;

76: a := band (ir, smask);                    {1111111 = DESP}
77: a := inv (a);
78: a := a + 1; goto 75;
```

Macroarquitetura
(ISA – *Instruction Set Architecture*)

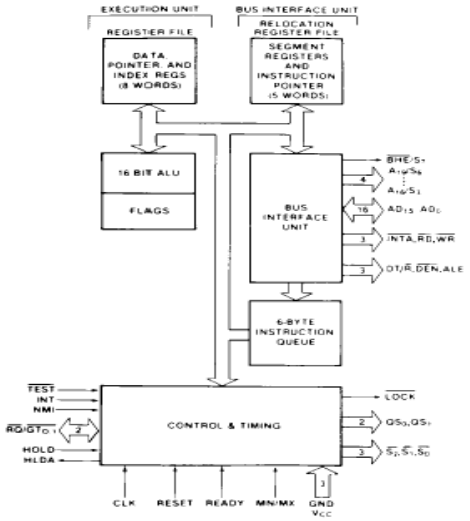
- Conjunto de Instruções
- Registradores
- Modelo de Memória
  - Endereçamento
  - Pilha
  - Memória Cache
  - Unidades Funcionais

Exemplos de Microarquiteturas

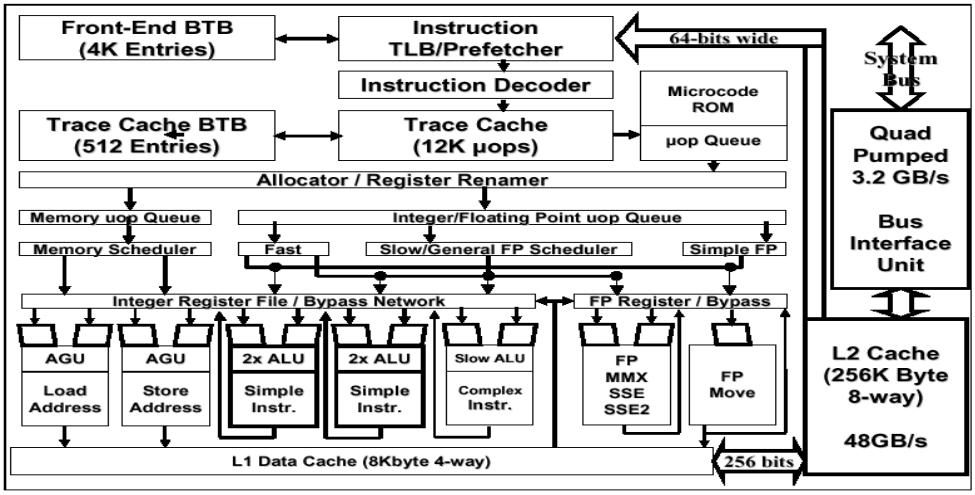
Arquitetura do 8051



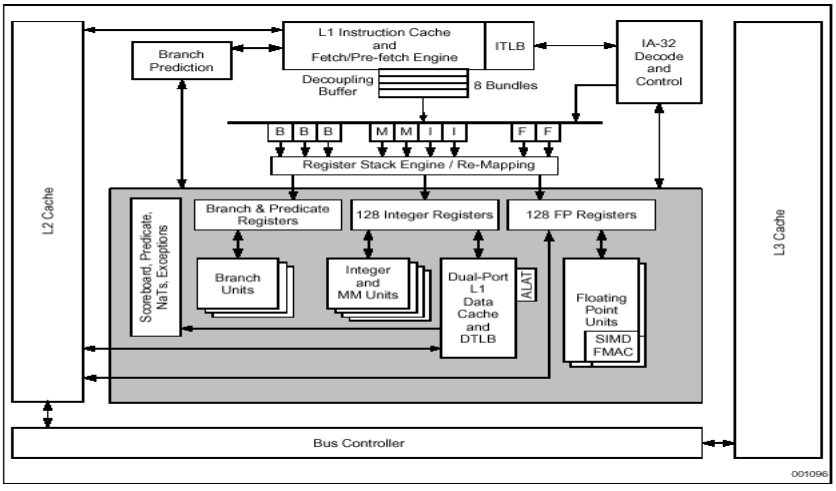
Arquitetura do 8086



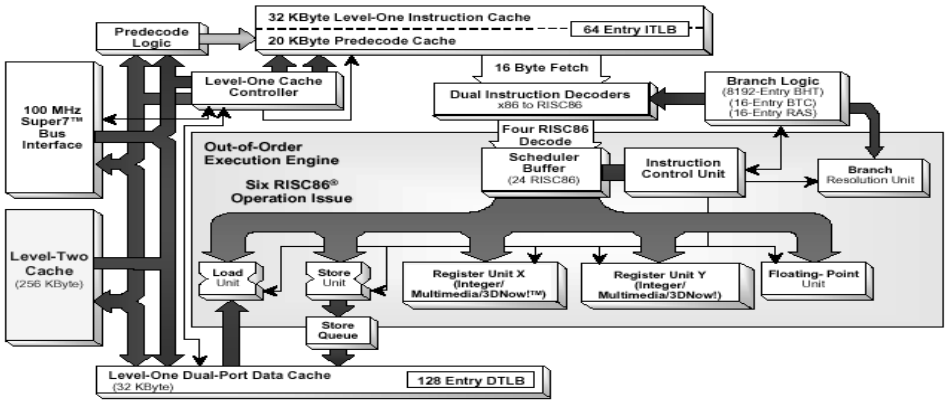
Arquitetura do Pentium 4



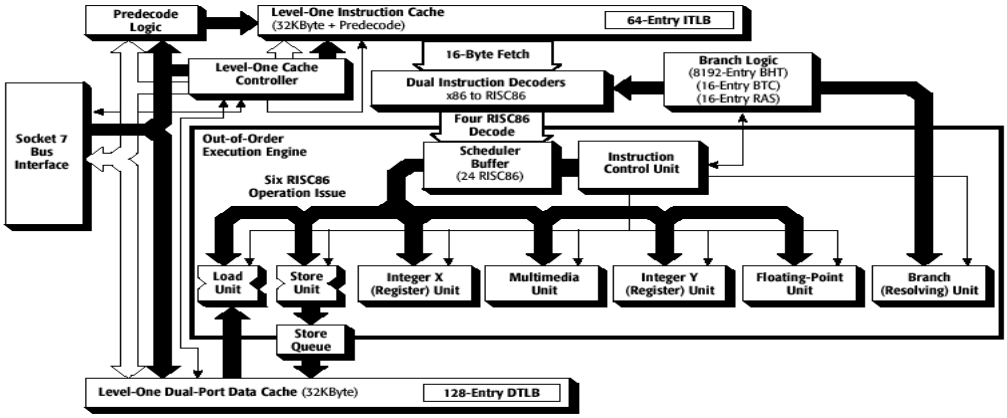
Arquitetura do Itanium



Arquitetura do AMD K6 - III



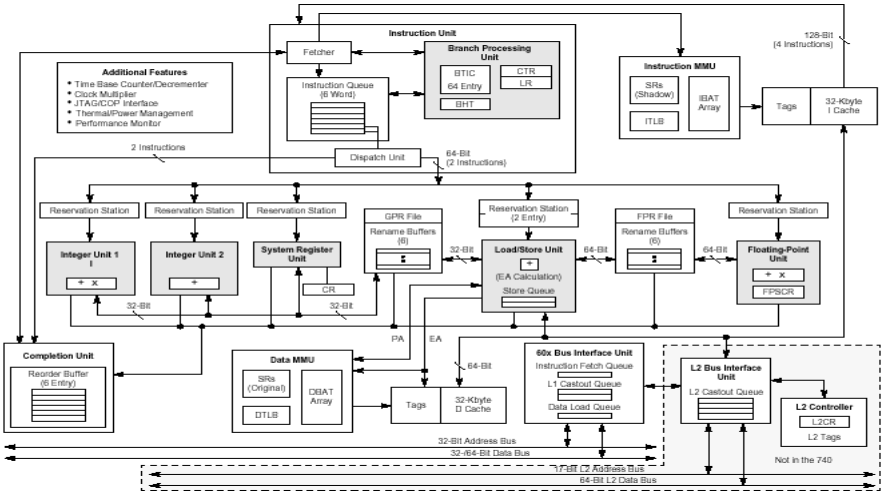
Arquitetura do AMD K6 - MMX



Professor
Edson Pedro FERLIN

Arquitetura de Computador

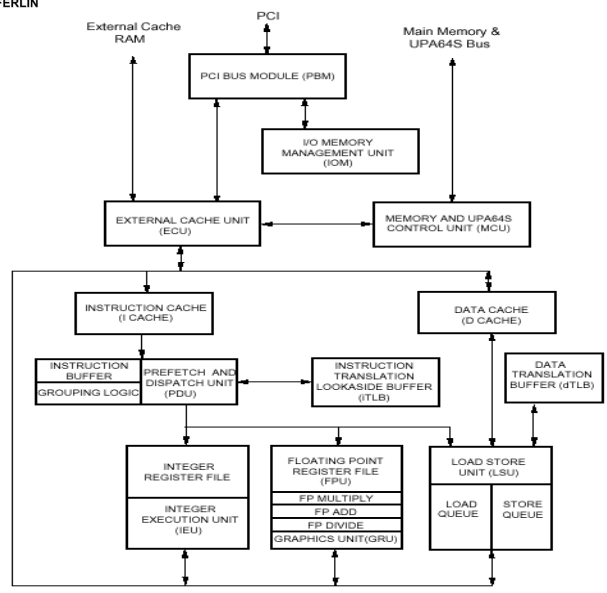
Arquitetura do PowerPC 750



Professor
Edson Pedro FERLIN

Arquitetura de Computador

Arquitetura do Ultra-SPARC III



Contato

eferlin@live.com

(BLOG) professorferlin.blogspot.com

(SITE) professorferlin.webnode.com.br

(YOUTUBE) ProfEdsonPedroFerlin