Estrutura do tema Avaliação de Desempenho (IA-32)

1. A avaliação de sistemas de computação (métricas)
2. Técnicas de otimização de *hardware*
    1. *Hierarquia de memória*
    2. *Exploração de paralelismo*
3. Técnicas de otimização de código (compiladores)
4. Outras técnicas de otimização (programador)
5. Medição de perfil de execução

Medição e otimização de desempenho
Impacto do nível de otimização (compilador)

gcc –m32 –O0 (sem otimizações) #I = 10 x N + …		-Os (opt para *code Size*)	
c7 45 f8 00 00 00 00	movl $0, -8(%ebp) # soma=0	31 c0	xorl %eax, %eax
c7 45 fc 00 00 00 00	movl $0, -4(%ebp) # i = 0	8b 55 08	movl 8(%ebp), %edx
…			
8b 45 fc	movl -4(%ebp), %eax	ff 44 c2 04	*incl 4(%edx,%eax,8)*
8d 14 c5 00 00 00 00	leal (,%eax,8), %edx # i*8	40	incl %eax
8b 45 08	movl 8(%ebp), %eax	83 f8 64	cmpl $100, %eax
01 d0	addl %edx, %eax # v+8*i	75 f6	jne -10
8b 50 04	*movl 4(%eax), %edx*		
83 c2 01	*addl $1, %edx # v[i].a++*		
89 50 04	*movl %edx, 4(%eax)*		
83 45 fc 01	addl $1, -4(%ebp) # i++	31 d2	xorl %edx, %edx
83 7d fc 63	cmpl $99, -4(%ebp)	f7 f1	divl %ecx
7e de	jle -34		
…			
ba 1f 85 eb 51	movl $1374389535, %edx		
…			

```c
struct S {
    char s[4]; unsigned int a;
};

void init(struct S *vec) {
    int i;
    for(i=0; i<N; i++) {
        vec[i].a = rand()%20;
    }
}

int media(struct S *vec) {
    int i;
    unsigned int soma=0;
    for (i=0 ; i<N ; i++)
        vec[i].a++;
    for (i=0 ; i<N ; i++)
        soma += vec[i].a;
    return(soma/N);
}

struct S v[N];

int main() {
    init(v);
    printf("media=%d\n", media(v) );
}
```

Medição do perfil de execução

Call-graph

- Tempo de cada chamada organizado de forma hierárquica

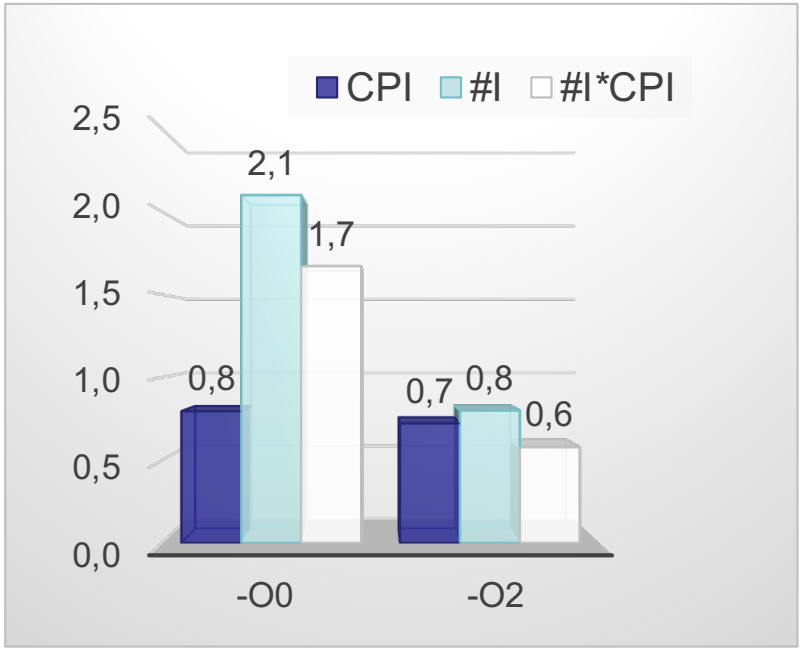
Time Profiler 〉 Profile 〉 Root

Weight	Self Weight	Symbol Name
4.76 s 100.0%	0 s	▼a.out (11436)
4.76 s 99.9%	0 s	▼start libdyld.dylib
4.76 s 99.9%	0 s	▼main a.out
2.51 s 52.6%	2.51 s	media a.out
2.20 s 46.1%	1.70 s	▼init a.out
500.00 ms 10.4%	500.00 ms	rand libsystem_c.dylib

- **clang-1000 -O0**
  - **Main**
    - **Init (46% do tempo de execução)**
      - **rand (10 %)**
    - **Media (53%) -> Hot-spot (ponto quente)**

Comparação -O0 vs -O2

(função **média** N= 10^8 - TPC8)



CPI - Ciclos Por Instrução (médio)

I - Instruções ($x10^9$)

I*CPI - Total de Ciclos ($x10^9$)

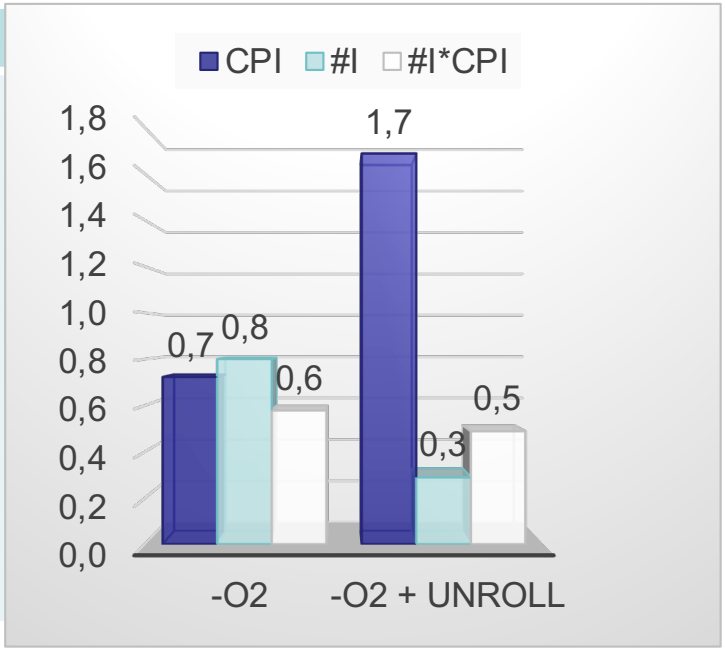
Medição e otimização de desempenho
Optimização Loop-unrolling

-O2

```
1e00:  83 00 01    addl $1, (%eax)
1e03:  83 c0 08    addl $8, %eax
1e06:  39 c1       cmpl %eax, %ecx
1e08:  75 f6       jne -10
```

-O2 -funroll-all-loops

```
1c90:  83 00 01       addl $1, (%eax)
1c93:  83 40 08 01    addl $1, 8(%eax)
1c97:  83 40 10 01    addl $1, 16(%eax)
1c9b:  83 40 18 01    addl $1, 24(%eax)
1c9f:  83 40 20 01    addl $1, 32(%eax)
1ca3:  83 40 28 01    addl $1, 40(%eax)
1ca7:  83 40 30 01    addl $1, 48(%eax)
1cab:  83 40 38 01    addl $1, 56(%eax)
1caf:  83 c0 40       addl $64, %eax
1cb2:  39 c1          cmpl %eax, %ecx
1cb4:  75 da          jne -38
```

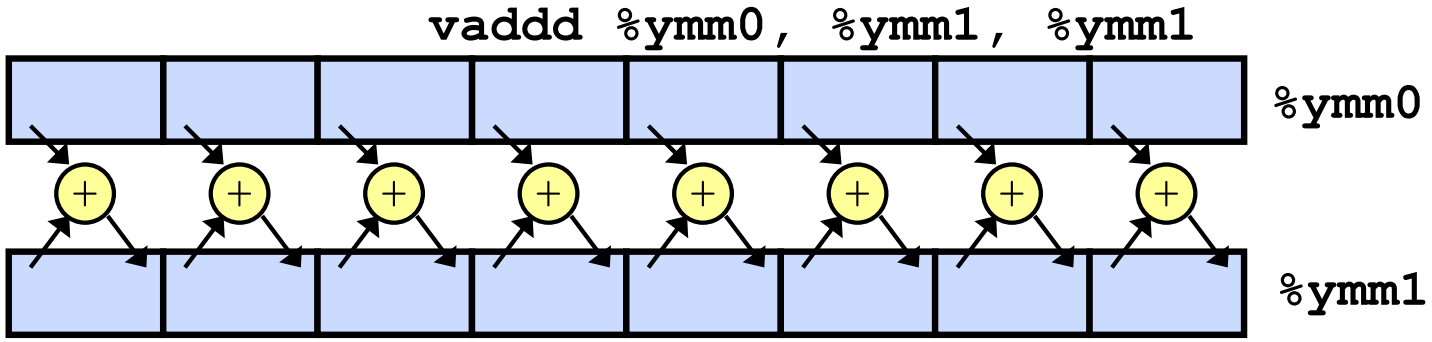


Ciclo desdobrado 8x

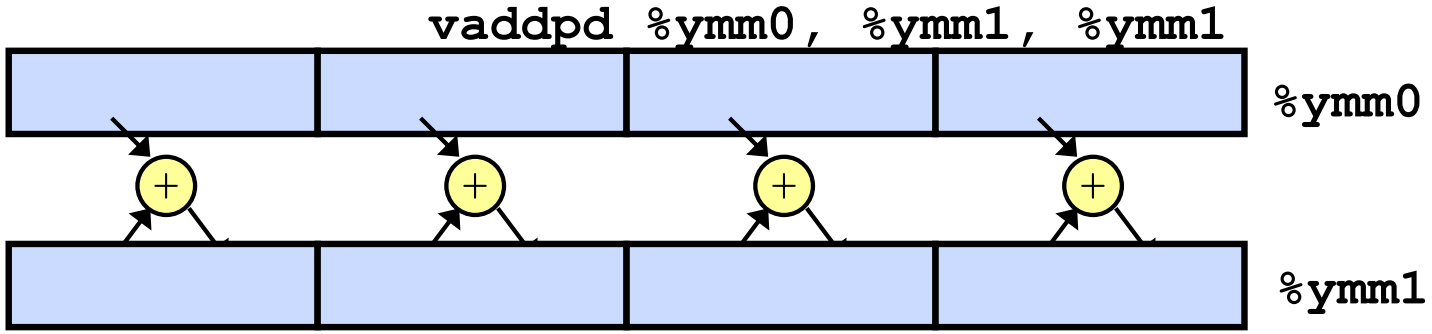
```
for(i=0;i<=N;i+=8) {
    v[i].a++;
    v[i+1].a++;
  ...
```

SIMD Operations (Vetorização)

- SIMD Operations: Int (8 x 4 bytes = 32 bytes)



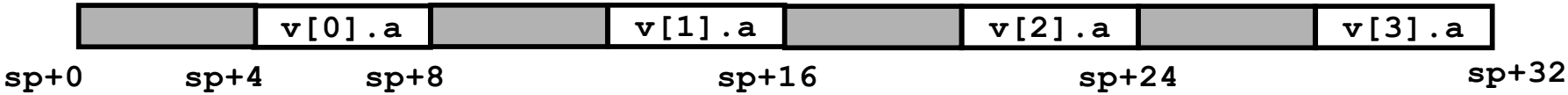
- SIMD Operations: Double Precision



Array de Estruturas vs Estruturas de Arrays

- **Problema:**
  - A função ***media*** acede a elementos que não estão armazenado em posições consecutivas da memória
    - Espaço de 4 bytes entre os elementos
    - Desperdiça largura de banda da memória

```
struct S {
  char s[4];
  unsigned int a;
} v[N];
```

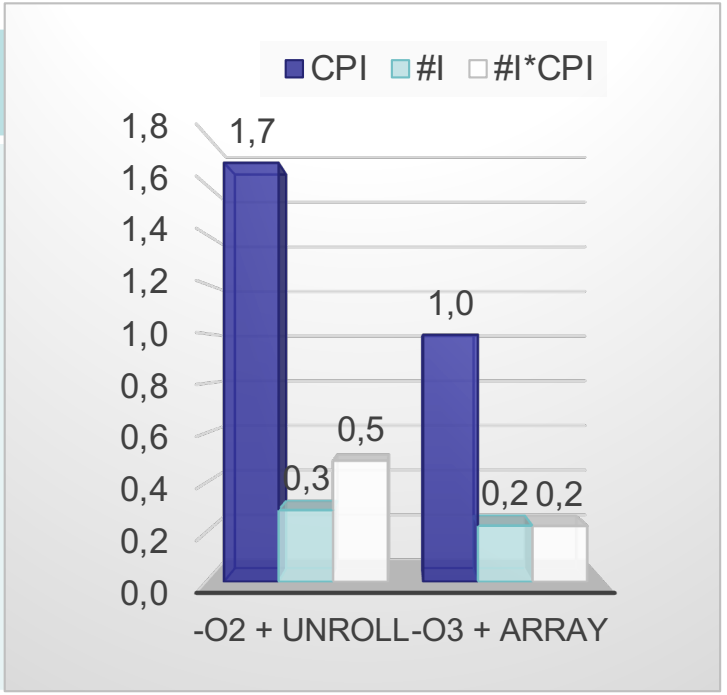


- Solução:
  - Passar para uma estrutura de com dois arrays
    - Nota: Código fica mais complexo

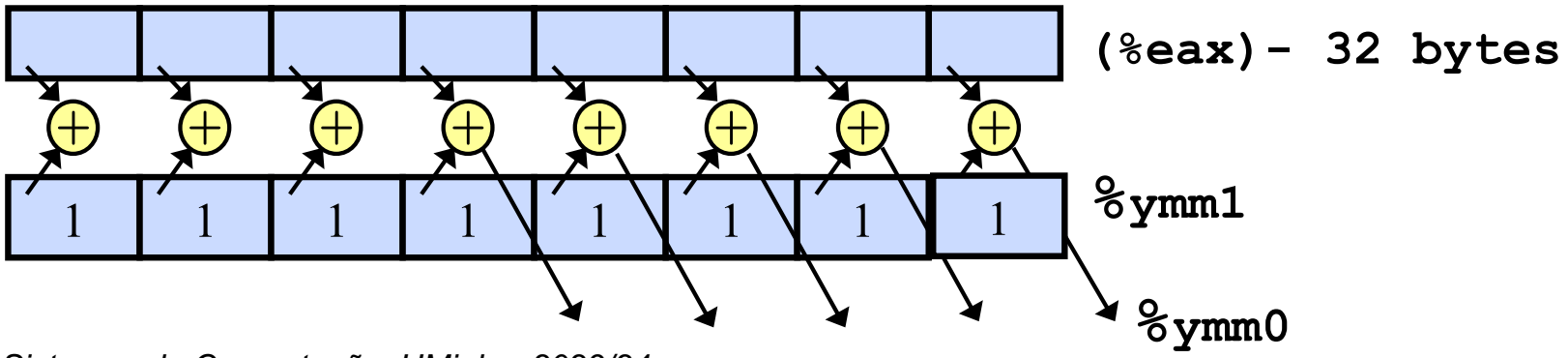
```
struct SoA {
  char s[4*N];
  unsigned int a[N];
} S;
```

Optimização Vetorização

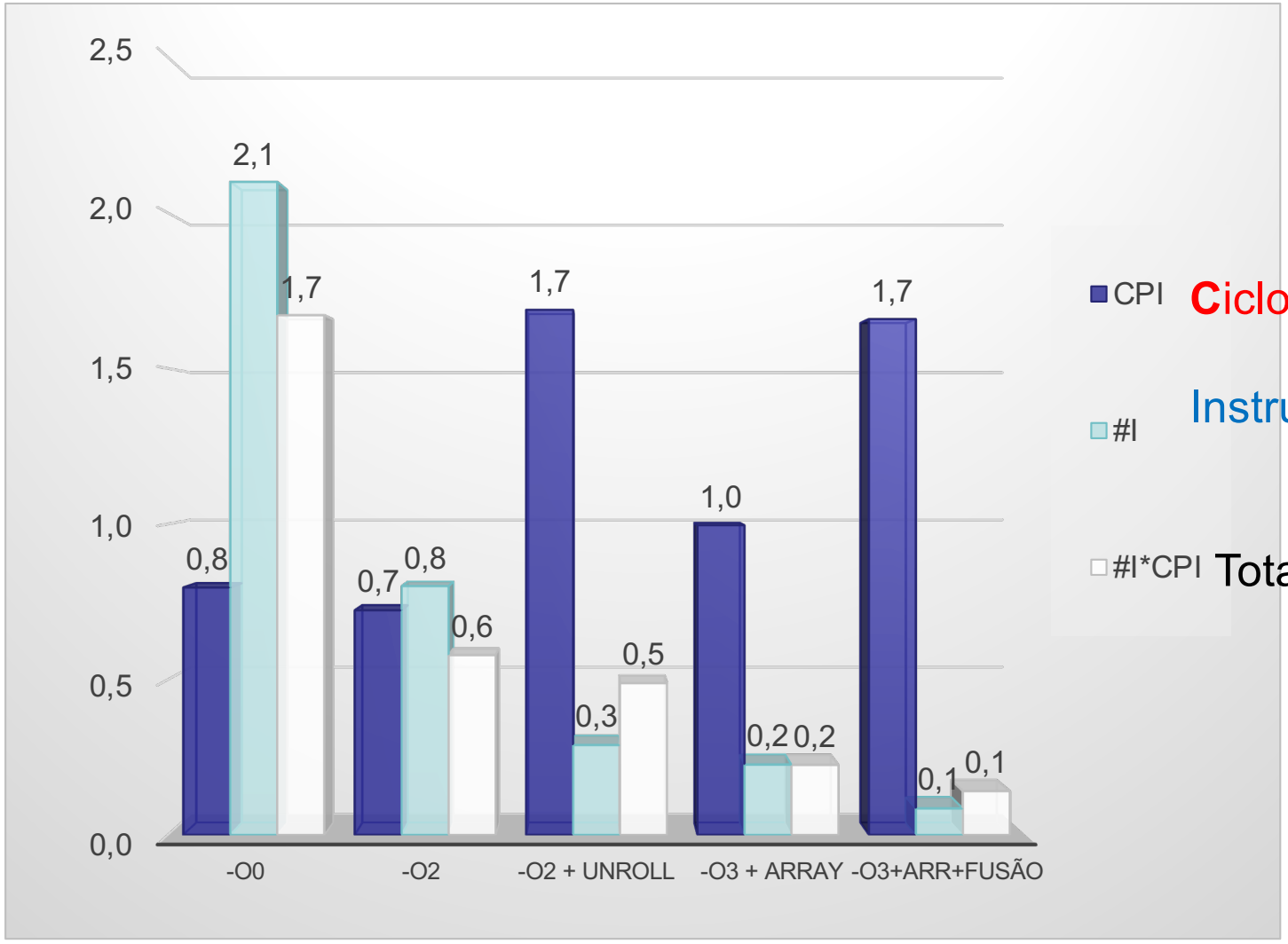
-O2 -funroll-all-loops			-O3 - código vetorizado - Array
1c90:	83 00 01	addl $1, (%eax)	L7:
1c93:	83 40 08 01	addl $1, 8(%eax)	**vpaddd (%eax), %ymm1, %ymm0**
1c97:	83 40 10 01	addl $1, 16(%eax)	addl $32, %eax
1c9b:	83 40 18 01	addl $1, 24(%eax)	vmovdqu %ymm0, -32(%eax)
1c9f:	83 40 20 01	addl $1, 32(%eax)	cmpl %ecx, %eax
1ca3:	83 40 28 01	addl $1, 40(%eax)	jne L7
1ca7:	83 40 30 01	addl $1, 48(%eax)	
1cab:	83 40 38 01	addl $1, 56(%eax)	processa 8 elementos de cada vez
1caf:	83 c0 40	addl $64, %eax	
1cb2:	39 c1	cmpl %eax, %ecx	
1cb4:	75 da	jne -38	



vpaddd (%eax), %ymm1, %ymm0



Medição e otimização de desempenho
Resumo – Comparação de tempos (#IxCPI)



Ciclos **P**or **I**nstrução (médio

Instruções ($x10^9$)

Total de Ciclos ($x10^9$)