CS 33: Introduction to Computer Organization

TA: Aalisha Dalal
LA: Jonathan Myong
Office Hours: Friday, 9:30-11:30AM

Outline

- **Compiler Optimization/ Limitations**
- **Passage to Parallelism**
- **Worksheet Problems**

Loop-Invariant/ Code Motion

```
int i = 0;
while (i < n) {
    x = y + z;
    a[i] = 6 * i + x * x;
    ++i;
}
```

How can this be optimized?

What remains unaffected by loop iterations?

Strength Reduction

```
1 unsigned bar(unsigned a) {
2     return a * 9 + 17;
3 }
```

```
1 leal    17(%rdi,%rdi,8), %eax
```

Share Common Expressions

```
/* Sum neighbors of i,j */
up =    val[(i-1)*n + j  ];
down =  val[(i+1)*n + j  ];
left =  val[i*n     + j-1];
right = val[i*n     + j+1];
sum = up + down + left + right;
```

```
long inj = i*n + j;
up =    val[inj - n];
down =  val[inj + n];
left =  val[inj - 1];
right = val[inj + 1];
sum = up + down + left + right;
```

```
leaq   1(%rsi), %rax  # i+1
leaq   -1(%rsi), %r8  # i-1
imulq  %rcx, %rsi     # i*n
imulq  %rcx, %rax     # (i+1)*n
imulq  %rcx, %r8      # (i-1)*n
addq   %rdx, %rsi     # i*n+j
addq   %rdx, %rax     # (i+1)*n+j
addq   %rdx, %r8      # (i-1)*n+j
...
```

```
imulq   %rcx, %rsi  # i*n
addq    %rdx, %rsi  # i*n+j
movq    %rsi, %rax  # i*n+j
subq    %rcx, %rax  # i*n+j-n
leaq    (%rsi,%rcx), %rcx # i*n+j+n
...
```

What prevents Compiler Optimization? - I

Procedure Calls

```
void lower(char *s)
{
  size_t i;
  for (i = 0; i < strlen(s); i++)
    if (s[i] >= 'A' && s[i] <= 'Z')
      s[i] -= ('A' - 'a');
}
```

What can be explicitly done to optimize the code?

Why Optimization is not possible? - I

Procedure Calls

```
void lower(char *s)
{
  size_t i;
  for (i = 0; i < strlen(s); i++)
    if (s[i] >= 'A' && s[i] <= 'Z')
      s[i] -= ('A' - 'a');
}
```

EXPLICIT

```
void lower(char *s)
{
  size_t i;
  size_t len = strlen(s);
  for (i = 0; i < len; i++)
    if (s[i] >= 'A' && s[i] <= 'Z')
      s[i] -= ('A' - 'a');
}
```

Why Optimization is not possible? - II

Memory Aliasing Example

```
# include <stdio.h>

int main()
{
 int arr[2] = { 1, 2 };
 int i=10;

 /* Write beyond the end of arr. Undefined behaviour in s
 arr[2] = 20;

 printf("element 0: %d \t", arr[0]); // outputs 1
 printf("element 1: %d \t", arr[1]); // outputs 2
 printf("element 2: %d \t", arr[2]); // outputs 20, if al
 printf("i: %d \t\t", i); // might also output 20, not 10
10
 /* arr size is still 2. */
 printf("arr size: %d \n", (sizeof(arr) / sizeof(int)));
}
```

Why Optimization is not possible? - II

```
/* Sum rows is of n X n matrix a
   and store in vector b  */
void sum_rows1(double *a, double *b, long n) {
    long i, j;
    for (i = 0; i < n; i++) {
        b[i] = 0;
        for (j = 0; j < n; j++)
            b[i] += a[i*n + j];
    }
}
```

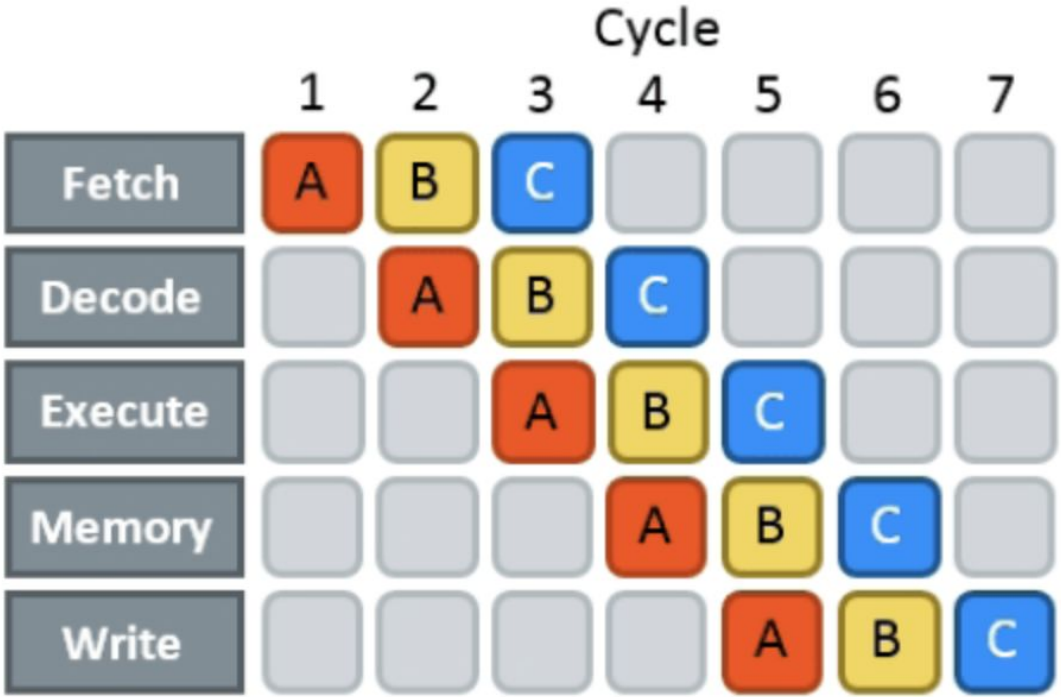
```
double A[9] =
  { 0,   1,   2,
    4,   8,  16,
   32,  64, 128};

double *B = A+3;

sum_rows1(A, B, 3);
```

```
double A[9] =
  { 0,   1,   2,
    3,  22, 224,
   32,  64, 128};
```

Concepts - Instruction Pipeline



Source: https://techdecoded.intel.io/resources/understanding-the-instruction-pipeline/

Latency vs. Throughput

An assembly line is manufacturing cars. It takes eight hours to manufacture a car and that the factory produces one hundred and twenty cars per day.

Latency is: 8 hours.

Throughput is: 120 cars / day or 5 cars / hour.

Latency vs. Throughput

Throughput - No. of instructions executed / cycle

Q: What are some ways to increase code throughput?

Latency - No. of cycles / instruction

Example - Sum Function

```
int sum = 0;
int i=0;
int n = 8;
int arr[n] = {5,4,3,2,1,6,7,9};

for(i=0; i<n;i=i+1)
{
    sum = sum + arr[i];
}
```

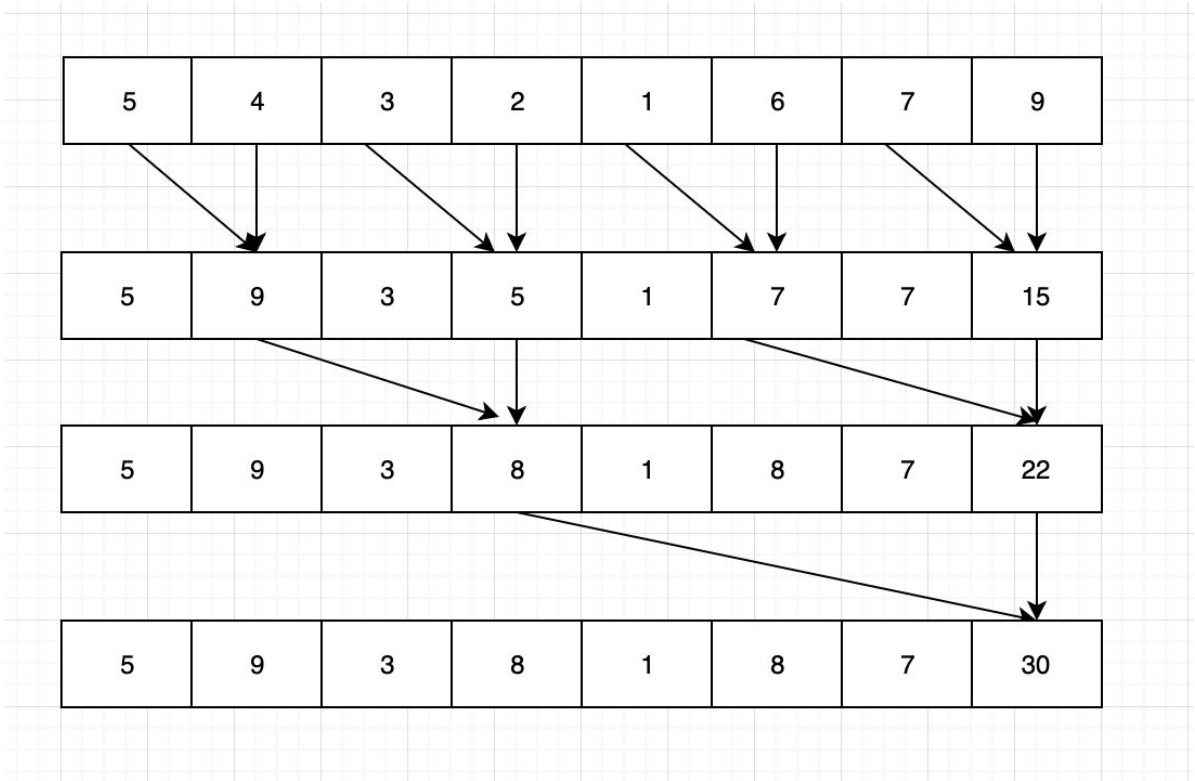
How can it be optimized?

Loop Unrolling - Sum Function

```
int sum = 0;
int i=0;
int n = 8;
int arr[n] = {5,4,3,2,1,6,7,9};

for(i=0; i<n;i=i+2)
{
    sum = sum + arr[i] + arr[i+1];
}
```

Optimal Parallelism Approach



Worksheet

https://tinyurl.com/cs33-ucla100