

# CS3350B Computer Organization

## Chapter 1: CPU and Memory

### Some Locality Examples

Iqra Batool

Department of Computer Science  
University of Western Ontario, Canada

Monday January 15, 2024

# Instruction Locality Example 1

A C function and program.

```
int doSomething() {  
    int z = 10+12;  
    return z;  
}  
  
int main() {  
    int a = 1;  
    int b = 2;  
    doSomething();  
    return 0;  
}
```

That program in assembly (MIPS).

```
doSomething:  
    add $t0 $0 $0  
    addi $t0 $t0 10  
    addi $t0 $t0 12  
    add $v0 $t0 $0  
    jr $ra  
  
main:  
    li $s0 1  
    li $s1 2  
    jal doSomething  
    li $v0 10  
    syscall
```

# Instruction Locality Example 1

That program in  
assembly (MIPS).

Program Binary (fake).

doSomething:

add \$t0 \$0 \$0

addi \$t0 \$t0 10

addi \$t0 \$t0 12

add \$v0 \$t0 \$0

jr \$ra

main:

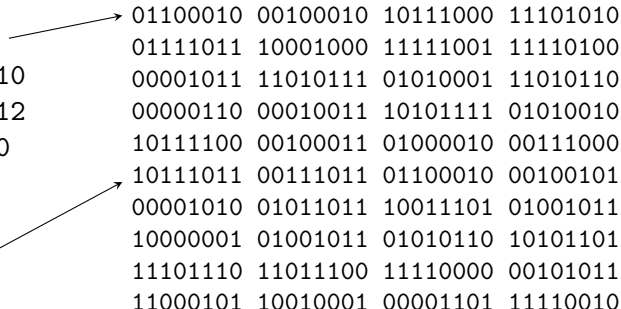
li \$s0 1

li \$s1 2

jal doSomething

li \$v0 10

syscall



01100010	00100010	10111000	11101010
01111011	10001000	11111001	11110100
00001011	11010111	01010001	11010110
00000110	00010011	10101111	01010010
10111100	00100011	01000010	00111000
10111011	00111011	01100010	00100101
00001010	01011011	10011101	01001011
10000001	01001011	01010110	10101101
11101110	11011100	11110000	00101011
11000101	10010001	00001101	11110010

A method call jumps to a different  
area of the program binary!


## Instruction Locality Example 2

An inlined C function and program.

```
inline int doSomething() {  
    int z = 10+12;  
    return z;  
}  
  
int main() {  
    int a = 1;  
    int b = 2;  
    doSomething();  
    return 0;  
}
```

That program in assembly (MIPS).

```
main:  
    li $s0 1  
    li $s1 2  
    add $t0 $0 $0  
    addi $t0 $t0 10  
    addi $t0 $t0 12  
    add $v0 $t0 $0  
    li $v0 10  
    syscall
```



## Instruction Locality Example 2

main:

li \$s0 1	
li \$s1 2	10011010 01110111 00101101 00110000
add \$t0 \$0 \$0	00010110 11011011 11000000 01101011
addi \$t0 \$t0 10	01001000 10011111 11010100 10100101
addi \$t0 \$t0 12	10000011 01101111 10110100 01100001
add \$v0 \$t0 \$0	00000101 01110010 00101010 00110101
li \$v0 10	00110000 00111010 11101001 11110010
syscall	10010000 00110001 11110000 00100000
	10101111 00110101 00100100 01110000

Inlined function  $\implies$  All instructions are sequential.

# Data Locality Example Without Arrays (1/2)

*Highly simplified example.*

Assume CPU has no registers and cache is 4 words using LRU.

```
int fibonacci1(int n) {  
    int t1 = 0, t2 = 1;  
    if (n < 1) {  
        return t1;  
    }  
    if (n < 2) {  
        return t2;  
    }  
  
    for (int i = 1; i < n ++i) {  
        int t3 = t1 + t2;  
        t1 = t2;  
        t2 = t3;  
    }  
    return t3;  
}
```

Cache Contents				Inst.	M/H
n				start	M
t1	n			t1 = 0	M
t2	t1	n		t2 = 1	M
n	t2	t1		n < 1; n < 2	H
i	n	t2	t1	i = 0; i < n	M; H
t1	t2	i	n	t1 + t2	H
t3	t1	t2	n	t3 = t1 + t2	M
t1	t2	t3	n	t1 = t2	H
t2	t3	t1	n	t2 = t3	H
i	t2	t3	t1	++i	M
i	n	t2	t3	i < n	M
t1	t2	i	n	t1 + t2	M
t3	t1	t2	n	t3 = t1 + t2	M
:					

- (1) "Longer" loop causes conditional to always miss
- (2) Initializing variables far away from use ruins cache of function caller.

## Data Locality Example Without Arrays (2/2)

*Highly simplified example.*

Assume CPU has no registers and cache is 4 words using LRU.

```
int fibonacci2(int n) {  
    if (n < 1) {  
        return 0;  
    }  
    if (n < 2) {  
        return 1;  
    }  
  
    int t1 = 0, t2 = 1;  
    for (int i = 1; i < n; ++i) {  
        t2 = t1 + t2;  
        t1 = t2 - t1;  
    }  
    return t2;  
}
```

Cache Contents				Inst.	M/H
n				start	-
n				n < 1; n < 2	H
t1	n			t1 = 0	M
t2	t1	n		t2 = 1	M
i	n	t2	t1	i = 0; i < n	M; H
t2	t1	i	n	t2 = t1 + t2	H
t1	t2	i	n	t1 = t2 - t1	H
i	t1	t2	n	++i	H
i	n	t1	t2	i < n	H
t2	t1	i	n	t2 = t1 + t2	H
t1	t2	i	n	t1 = t2 - t1	H
:					

- (1) No excess variables in loop; everything in cache.
- (2) Local variables don't ruin caller's cache in degenerative cases (n already in cache of caller).