ECGR 5181  
Assignment 4: Pipelined CPU RAM SingleCore & MultiCore Simulation

Patrick Flynn  
Naseeruddin Lodge  
Nahush Tambe

Description

The CPI is low for part 1 since we have a direct connection to memory. Therefore, there is very little overhead and the stalls are minimal since only one program is running. The CPI extends for part 2 since we have the bus. The bus on its own increases the CPI since you have the extra layer between the CPU and memory, but since both CPUs read and write from the same places in memory, there are stalls as the bus can only handle one request at the time- that is, if two CPUs request the same memory address, the bus can only handle one request at a time.

We are able to handle this because we use classes, so creating new CPUs just requires us to instantiate the new classes. Once we create these classes, we use a common RAM and bus class and assign them to each CPU, and then we use C++ threads to run each CPU.

PART I

VADD

; x2 is the stack sp

addi x2, x0, 512 -> 1 Cycle

; i

add x6, x0, x0 -> 1 Cycle

sw x6, 4(x2) -> 2 Cycles

add x6, x0, x0 -> 1 Cycle

; 256

addi x7, x0, 256 -> 1 Cycle

loop\_cmp:

lw x25, 4(x2) -> 2 Cycles

add x0, x0, x0 -> 1 Cycle

add x0, x0, x0 -> 1 Cycle

bge x25, x7, loop\_done -> 1 Cycle

loop:

lw x6, 4(x2) -> 2 Cycles

add x0, x0, x0 -> 1 Cycle

srli x20, x6, 2 -> 1 Cycle

flw f4, 1024(x20) -> 2 Cycles

flw f5, 2048(x20) -> 2 Cycles

add x0, x0, x0 -> 1 Cycle

add x0, x0, x0 -> 1 Cycle

fadd.s f6, f4, f5 -> 1 Cycle

fsw f6, 3072(x20) -> 2 Cycle

lw x21, 4(x2) -> 2 Cycles

add x0, x0, x0 -> 1 Cycle

add x0, x0, x0 -> 1 Cycle

addi x21, x21, 1 -> 1 Cycle

sw x21, 4(x2) -> 2 Cycles

add x0, x0, x0 -> 1 Cycle

beq x0, x0, loop\_cmp -> 1 Cycle

loop\_done:

addi x30, x0, 20 -> 1 Cycle

Total Clock Cycles -> 1 + 1+2+1 + 1 + (2+1+1+1 + 2+1+1 + 2+2+1+1 + 1 + 2 + 2+1+1+1+2+1+1)\*256 + 1

-> 6 + 256\*(27) + 1

-> 6919 Clock Cycles

Clock Cycles per Instruction -> Total Clock Cycles / Instruction Count

-> 6919 / (5 + (20)\*256 + 1)

-> 6919 / 5126

-> 1.350 CPI

PART II

CPU 0

Vadd  
  
; x2 is the stack sp

addi x2, x0, 512 -> 1 Cycle

; i

add x6, x0, x0 -> 1 Cycle

sw x6, 4(x2) -> 2 Cycles

add x6, x0, x0 -> 1 Cycle

; 256

addi x7, x0, 256 -> 1 Cycle

loop\_cmp:

lw x25, 4(x2) -> 2 Cycles

add x0, x0, x0 -> 1 Cycle

add x0, x0, x0 -> 1 Cycle

bge x25, x7, loop\_done -> 1 Cycle

loop:

lw x6, 4(x2) -> 2 Cycles

add x0, x0, x0 -> 1 Cycle

srli x20, x6, 2 -> 1 Cycle

flw f4, 1024(x20) -> 2 Cycles

flw f5, 2048(x20) -> 2 Cycles

add x0, x0, x0 -> 1 Cycle

add x0, x0, x0 -> 1 Cycle

fadd.s f6, f4, f5 -> 5 Cycle

fsw f6, 3072(x20) -> 2 Cycle

lw x21, 4(x2) -> 2 Cycles

add x0, x0, x0 -> 1 Cycle

add x0, x0, x0 -> 1 Cycle

addi x21, x21, 1 -> 1 Cycle

sw x21, 4(x2) -> 2 Cycles

add x0, x0, x0 -> 1 Cycle

beq x0, x0, loop\_cmp -> 1 Cycle

loop\_done:

addi x30, x0, 20 -> 1 Cycle

Total Clock Cycles -> 1 + 1+2+1 + 1 + (2+1+1+1 + 2+1+1 + 2+2+1+1 + 5 + 2 + 2+1+1+1+2+1+1)\*256 + 1

-> 6 + 256\*(31) + 1

-> 7943 Clock Cycles

Clock Cycles per Instruction -> Total Clock Cycles / Instruction Count

-> 7943 / (5 + (20)\*256 + 1)

-> 7943 / 5126

-> 1.550 CPI

CPU 1

VSUB  
  
; x2 is the stack sp

addi x2, x0, 768 -> 1 Clock Cycle

; i

add x6, x0, x0 -> 1 Clock Cycle

sw x6, 4(x2) -> 2 Clock Cycles

add x6, x0, x0 -> 1 Clock Cycle

; 256

addi x7, x0, 256 -> 1 Clock Cycle

loop\_cmp:

lw x25, 4(x2) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

add x0, x0, x0 -> 1 Clock Cycle

bge x25, x7, loop\_done -> 1 Clock Cycle

loop:

lw x6, 4(x2) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

srli x20, x6, 2 -> 1 Clock Cycle

flw f4, 1024(x20) -> 2 Clock Cycles

flw f5, 2048(x20) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

add x0, x0, x0 -> 1 Clock Cycle

fsub.s f6, f4, f5 -> 5 Clock Cycles

lui x17, 1 -> 1 Clock Cycle

add x18, x20, x17 -> 1 Clock Cycle

fsw f6, 0(x18) -> 2 Clock Cycles

lw x21, 4(x2) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

add x0, x0, x0 -> 1 Clock Cycle

addi x21, x21, 1 -> 1 Clock Cycle

sw x21, 4(x2) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

beq x0, x0, loop\_cmp -> 1 Clock Cycle

loop\_done:

addi x30, x0, 20 -> 1 Clock Cycle

Total Clock Cycles -> 1 + 1+2+1 + 1 + (2+1+1+1 + 2+1+1 + 2+2+1+1 + 5 + 1+1+2 + 2+1+1+1+2+1+1)\*256 + 1

-> 6 + 256\*(33) + 1

-> 8455 Clock Cycles

Clock Cycles per Instruction -> Total Clock Cycles / Instruction Count

-> 8455 / (5 + (20)\*256 + 1)

-> 8455 / 5126

-> 1.649 CPI

CPU 0 + CPU 1

CPU0

Vadd

; x2 is the stack sp

addi x2, x0, 512 -> 1 Clock Cycle

; i

add x6, x0, x0 -> 1 Clock Cycle

sw x6, 4(x2) -> 2 Clock Cycles

add x6, x0, x0 -> 1 Clock Cycle

; 256

addi x7, x0, 256 -> 1 Clock Cycle

loop\_cmp:

lw x25, 4(x2) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

add x0, x0, x0 -> 1 Clock Cycle

bge x25, x7, loop\_done -> 1 Clock Cycle

loop:

lw x6, 4(x2) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

srli x20, x6, 2 -> 1 Clock Cycle

flw f4, 1024(x20) -> 2 Clock Cycles

flw f5, 2048(x20) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

add x0, x0, x0 -> 1 Clock Cycle

fadd.s f6, f4, f5 -> 5 Clock Cycle

fsw f6, 3072(x20) -> 2 Clock Cycle

lw x21, 4(x2) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

add x0, x0, x0 -> 1 Clock Cycle

addi x21, x21, 1 -> 1 Clock Cycle

sw x21, 4(x2) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

beq x0, x0, loop\_cmp -> 1 Clock Cycle

loop\_done:

addi x30, x0, 20 -> 1 Clock Cycle

CPU1

Vsub

; x2 is the stack sp

addi x2, x0, 768 -> 1 Clock Cycle

; i

add x6, x0, x0 -> 1 Clock Cycle

sw x6, 4(x2) -> 2 Clock Cycles

add x6, x0, x0 -> 1 Clock Cycle

; 256

addi x7, x0, 256 -> 1 Clock Cycle

loop\_cmp:

lw x25, 4(x2) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

add x0, x0, x0 -> 1 Clock Cycle

bge x25, x7, loop\_done -> 1 Clock Cycle

loop:

lw x6, 4(x2) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

srli x20, x6, 2 -> 1 Clock Cycle

flw f4, 1024(x20) -> 2 Clock Cycles

flw f5, 2048(x20) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

add x0, x0, x0 -> 1 Clock Cycle

fsub.s f6, f4, f5 -> 5 Clock Cycles

lui x17, 1 -> 1 Clock Cycle

add x18, x20, x17 -> 1 Clock Cycle

fsw f6, 0(x18) -> 2 Clock Cycles

lw x21, 4(x2) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

add x0, x0, x0 -> 1 Clock Cycle

addi x21, x21, 1 -> 1 Clock Cycle

sw x21, 4(x2) -> 2 Clock Cycles

add x0, x0, x0 -> 1 Clock Cycle

beq x0, x0, loop\_cmp -> 1 Clock Cycle

loop\_done:

addi x30, x0, 20 -> 1 Clock Cycle

Total Clock Cycles -> 1 + 1+2+1 + 1 + (2+1+1+1 + 2+1+1 + 2+2+1+1 + 5 + 2 + 2+1+1+1+2+1+1)\*256 + 1 + 1 + 1+2+1 + 1 + (2+1+1+1 + 2+1+1 + 2+2+1+1 + 5 + 1+1+2 + 2+1+1+1+2+1+1)\*256 + 1

-> 6 + 256\*(31) + 1 + 6 + 256\*(33) + 1

-> 7943 + 8455

-> 16398

Clock Cycles per Instruction -> Total Clock Cycles / Instruction Count

-> 16398 / ( (5 + (20)\*256 + 1) + (5 + (20)\*256 + 1) )

-> 16398 / ( 5126 + 5126 )

-> 16398 / 10252

-> 1.599 CPI