Homework 2 Review

Cornell CS 3410

Calling Conventions

You are given the following function in C, and you know that the function mod exists.

```
int gcd(int x, int y) {
  int t, a=x, b=y;
  while(b != 0) {
    t = b;
    b = mod(a,b);
    a = t;
  }
  return a;
}
```

Calling Conventions (Contd.)

How many caller/callee save registers will you use and for which variables?

Calling Conventions (Contd.)

How many caller/callee save registers will you use and for which variables?

1 callee save register for t, which is needed after a function call

2 caller save registers for a and b

Calling Conventions (Contd.)

Write the prologue for gcd().

Calling Conventions (Contd.)

Write the prologue for gcd().

```
addiu $sp, $sp, -28
sw $ra, 24($sp)
sw $fp, 20($sp)
sw $s0, 16($sp)
addiu $fp, $sp, 24
addiu $t0, $a0, 0
addiu $t1, $a1, 0
```

Calling Conventions (Contd.)

Write the epilogue for gcd().

Calling Conventions (Contd.)

Write the epilogue for gcd().

```
lw $s0, 16($sp)
lw $fp, 20($sp)
lw $ra, 24($sp)
addiu $sp, $sp, 28
jr $ra
nop
```

Calling Conventions (Contd.)

Write the function call for mod().

```
addiu $a0, $t0, 0
addiu $a1, $t1, 0
jal MOD
nop
```

Virtual Memory

Consider a byte addressable virtual memory system with 32 bit virtual addresses, 32-bit physical addresses, and 1 KB pages.

Virtual Memory (Contd.)

Is it possible to run a program that requires 16 GB of data on the system?

The amount of virtual memory is 2^32 = 4GB, and the system has 2^32 = 4GB physical memory.

Since only 4GB of memory are addressable, it is NOT possible to run such a program.

Virtual Memory (Contd.)

If each entry in the single-level page table is 4 bytes long, how much space does the page-table occupy in memory?

Virtual Memory (Contd.)

If each entry in the single-level page table is 4 bytes long, how much space does the page-table occupy in memory?

The page size is 1KB = 2^10 bytes, so the page-offset is 10 bits long. The VPN field is 32-10 = 22 bits long. The single-level page table thus has 2^22 entries. So, the table takes up 4*2^22 bytes = 16MB in memory.

Virtual Memory (Contd.)

If each entry in the single-level page table is 4 bytes long, how much total physical memory would a 32MB process occupy?

Virtual Memory (Contd.)

If each entry in the single-level page table is 4 bytes long, how much total physical memory would a 32MB process occupy?

Total physical memory = 16MB + 32MB = 48MB

Multicore Performance

Suppose we have a program of which 10% can be parallelized to any extent, 30% can be parallelized to run on up to 10 processors, and 60% can’t be parallelized at all. The program takes 70 seconds on a single processor.

Multicore Performance (Contd.)

How long will it take to run on 2 processors?

N as N goes to infinity?

Multicore Performance (Contd.)

How long will it take to run on 2 processors?

(70 * .6) + (70 * .1)/2 + (70 * .3)/2

= 42 + 3.5 + 10.5 = 56 seconds

N as N goes to infinity?

$$\lim_{N\to\infty}(70 \times 0.6) + \left(\frac{70*0.1}{N}\right) + \left(\frac{70*0.3}{\max(N,10)}\right) = 42 + 0 + 2.1 = 44.1 \text{ seconds}$$

Hazards, Branches & Cycle charts

```
        xor   t0, t0, t0
        beq   t0, r0, L1
        add   t1, t2, t3
        add   t2, t3, t1
L1:     add   t1, t1, t2
```

Hazards, Branches & Cycle charts

```
        xor   t0, t0, t0
        beq   t0, r0, L1
        add   t1, t2, t3
        add   t2, t3, t1
L1:     add   t1, t1, t2
```

Is the branch taken?

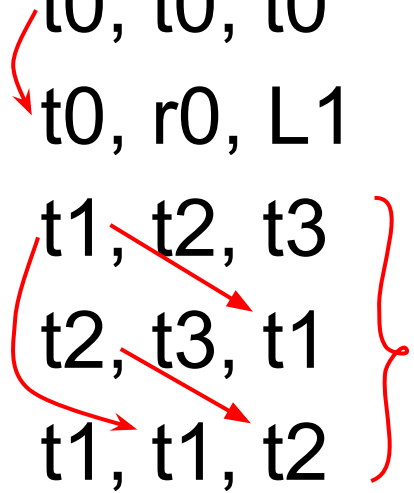
Hazards, Branches & Cycle charts

```
      xor   t0, t0, t0
      beq   t0, r0, L1
      add   t1, t2, t3
      add   t2, t3, t1
L1:   add   t1, t1, t2
```

Data hazards?

Hazards, Branches & Cycle charts

```
        xor   t0, t0, t0
        beq   t0, r0, L1
        add   t1, t2, t3
        add   t2, t3, t1
L1:     add   t1, t1, t2
```



Depends on branch & delay slot

Data hazards?

1 delay slot, predict not-taken, no bypasses, resolve branch in EX

xor t0, t0, t0														
beq t0, r0, L1														
add t1, t2, t3														
add t2, t3, t1														
add t1, t1, t2														

1 delay slot, predict not-taken, no bypasses, resolve branch in EX

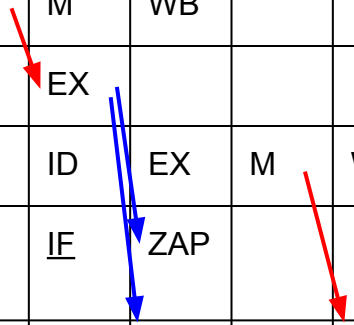
xor	t0, t0, t0	IF	ID	EX	M	WB								
beq	t0, r0, L1		IF	ID	ID	ID	ID	EX						
add	t1, t2, t3			IF	IF	IF	IF	ID	EX	M	WB			
add	t2, t3, t1							IF	ZAP					
add	t1, t1, t2								IF	ID	ID	ID	EX	M	WB

1 delay slot, predict not-taken, fully bypassed, resolve branch in EX

xor	t0, t0, t0													
beq	t0, r0, L1													
add	t1, t2, t3													
add	t2, t3, t1													
add	t1, t1, t2													

1 delay slot, predict not-taken, fully bypassed, resolve branch in EX

xor	t0, t0, t0	IF	ID	EX	M	WB								
beq	t0, r0, L1		IF	ID	EX									
add	t1, t2, t3			IF	ID	EX	M	WB						
add	t2, t3, t1				IF	ZAP								
add	t1, t1, t2					IF	ID	EX	M	WB				



Cache

32-bit machine, 16-byte cache lines, direct mapped, 1024 cache lines.

tag bits?

index bits?

offset bits?

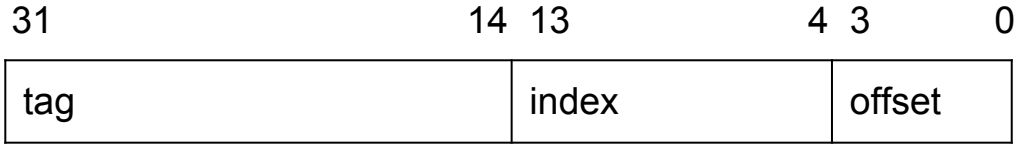
Cache

32-bit machine, 16-byte cache lines, direct mapped, 1024 cache lines.

tag bits? 18

index bits? 10

offset bits? 4



Cache

LW 0x0

LW 0x8

LW 0x100

LW 0x10000

LW 0x0

Cache

LW 0x0	?	tag=? index=?
LW 0x8	?	tag=? index=?
LW 0x100	?	tag=? index=?
LW 0x10000	?	tag=? index=?
LW 0x0	?	tag=? index=?

Cache

LW 0x0	Miss	tag=0 index=0
LW 0x8	Hit	tag=0 index=0
LW 0x100	Miss	tag=0 index=16
LW 0x10000	Miss	tag=4 index=0
LW 0x0	Miss	tag=0 index=0