

Computer Science 210
Computer Systems 1
2007 Semester 1
Lecture Notes

Load/Store Instructions

Lecture 5
24 Mar 07

James Goodman



Recommended Readings

- Today's lecture mostly based on Section 4.2 of Dr. Hutton's notes.

29-Mar-07

CS210

2

Views of Memory

- An array of bytes

000	
001	
002	
003	
004	
005	
006	
007	
008	
009	
00a	
00b	
00c	
00d	
00e	
00f	
010	
011	
...	

29-Mar-07

CS210

3

Views of Memory

- An array of words

000		
002		
004		
006		
008		
00a		
00c		
00e		
010		
012		
014		
016		
018		
01a		
01c		
01e		
020		
022		
...		

29-Mar-07

CS210

4

Views of Memory

- An array of longwords

000				
004				
008				
00c				
010				
014				
018				
01c				
020				
024				
028				
02c				
030				
034				
038				
03c				
040				
044				
...				

Views of Memory

- An array of quadwords

000							
008							
010							
018							
020							
028							
030							
038							
040							
048							
050							
058							
060							
068							
070							
078							
080							
088							
...							

Which is Correct View of Memory?

000		000			000				
001		002			004				
002		004			008				
003		006			00c				
004		008			010				
005		00a			014				
006		00c			018				
007		00e			01c				
008		010			020				
009		012			024				
00a		014			028				
00b		016			02c				
00c		018			030				
00d		01a			034				
00e		01c			038				
00f		01e			03c				
010		020			040				
011		022			044				
...		...			...				

Byte Order

- Little Endian

		1	0			11	10	01	00
000		000				000			
001		002				004			
002		004				008			
003		006				00c			
004		008				010			
005		00a				014			
006		00c				018			
007		00e				01c			
008		010				020			
009		012				024			
00a		014				028			
00b		016				02c			
00c		018				030			
00d		01a				034			
00e		01c				038			
00f		01e				03c			
010		020				040			
011		022				044			
...		...				...			

Byte Order

- Big Endian

		0	1		00	01	10	11
000		000		000				
001		002		004				
002		004		008				
003		006		00c				
004		008		010				
005		00a		014				
006		00c		018				
007		00e		01c				
008		010		020				
009		012		024				
00a		014		028				
00b		016		02c				
00c		018		030				
00d		01a		034				
00e		01c		038				
00f		01e		03c				
010		020		040				
011		022		044				
...		...		...				

Alignment

- An array of words (little-endian)

000		
002		
004		
006		
008		
00a		
00c		
00e		
010		
012		
014		
016		
018		
01a		
01c		
01e		
020		
022		
...		

Aligned word (008) →

Unaligned word (01f) →

Views of Memory

- An array of longwords (little-endian)

000				
004				
008				
00c				
010				
014				
018				
01c				
020				
024				
028				
02c				
030				
034				
038				
03c				
040				
044				
...				

Aligned longword (008) →

Unaligned longword (01f) →

Views of Memory

- An array of quadwords (little-endian)

000							
008							
010							
018							
020							
028							
030							
038							
040							
048							
050							
058							
060							
068							
070							
078							
080							
088							
...							

Aligned quadword (008) →

Unaligned quadword (01f) →

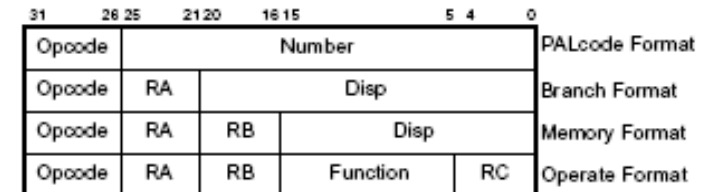
Load Instructions

- **ldq reg, ...** Load quadword
- **ldl reg, ...** Load *sign-extended* longword
- **ldwu reg, ...** Load *zero-extended* word
- **ldbu reg, ...** Load *zero-extended* byte

- **stq reg, ...** Store quadword
- **stl reg, ...** Store longword
- **stw reg, ...** Store word
- **stb reg, ...** Store byte

Motivation: Instruction Formats

Figure 1-1: Instruction Format Overview



– From The Alpha Architecture Handbook, Compaq Computer Corporation, 1998.

Load & Stores

ldq Reg, Address ! Direct

What's the problem?

- Address is stored as a constant inside the instruction
 - How to *dynamically* change the address?
- Instructions should be small, fixed size
 - Addresses are large
 - How to store a 51-bit address in a 32-bit instruction?
- Also a problem for branch instructions:

bne Reg, Address

Where Do the Bits Go?

Opcode	Src1	Src2	Dest
--------	------	------	------

- Register specification requires 5 bits
- Memory specification requires up to 51 bits (at least 43)
- Opcode specification requires ?
 - 515 unique opcodes
 - 10 bits required?

Arithmetic/Logical Instruction

Operate	Src1	Src2	Function	Dest
6 bits	5 bits	5 bits	11 bits	5 bits

- Opcode indicates a class of instructions
 - Opcode 10_{16} indicates an arithmetic function
 - Opcode 11_{16} indicates a logical function
 - Opcode 12_{16} indicates a shift function
- “Function” is extended Opcode, specifying which arithmetic/logical function

Example: addq

01 0000	Src1	Src2	000 0010 0000	Dest
6 bits	5 bits	5 bits	11 bits	5 bits

- Opcode for arithmetic instructions is 10_{16}
- Function code for addq instruction is 20_{16}
- Src1, Src2 and Dest each specify 1 of 32 registers

Example: bne

11 1010	Reg	Target
6 bits	5 bits	21 bits ???

- Opcode for bne instruction is $3d_{16}$
- Test register specifies 1 of 32 registers for testing
- Only 21 bits left for target instruction address!!!

Load/Store Instructions

Load/Store	Reg	Target
6 bits	5 bits	21 bits

- How to specify memory address (*effective address*) with only 21 bits?