Cheatography

cs2100 midterms Cheat Sheet

by jackiechenjx via cheatography.com/194887/cs/40724/

Number System

	1s-complement	2s-complement
Min	$-(2^{n-1}-1)$	$-(2^{n-1})$
Max	$2^{n-1}-1$	$2^{n-1}-1$
Zero(s)	0, 2^n-1	0
Negation	2^n - X -1	2^n-X
Addition[1]	Add carry out to result	Ignore carry out
	Sign-Magnitude	Excess
Min	$-(2^{n-1}-1)$	-excess
Max	$2^{n-1}-1$	2^n-excess-1
Zero(s)	0, 2^{n-1}	excess
	Diminished radix (r-1)'s complement	Radix r's complement
Definition[2]	(r^n - 1) - X	r^n - X

ASCII: 1bit parity + 7bit content

C string: Should end with '\0'

[1] overflow if the result is opposite sign of A and B

[2] radix -> base (e.g. 999...99 - X for base-9)

MIPS - Assembly

branch/jump labels	Does not count as instn.
lw & sw	Should use offset in multiple of 4
Instruction Executed	Initial instn. +num of loops * loop instn. + exit loop instn. + end instn.

MIPS - Memory

32 registers, each 32-bit (4-byte) long

Each word contains 32 bits (4 bytes)

Memory addresses are 32-bit long

2^{30} memory words[1]

[1] Consecutive words differ by 4 bytes

MIPS - Encoding

branch	If the branch is **not taken**:
	PC=PC+4
	If the branch is **taken**:
	PC=(PC+4)+immd*4

I-format[1]	16-bit immd only!
J-format[2]	26-bit target address
	Max Jump: 256MB

[1] `li` = `lui` upper 16-bit + `ori` lower 16-bit

[2] Actual address: 4-bit PC MSB + 26-bits + 2-bit word-aligned(00)

ISA - Design Philosophy

RISC	CISC
E.g. x86-32	E.g. MIPS, ARM
Single instruction performs complex operation	Small and simple instruction set
Smaller program size as memory was premium	Complex implementation, no room for hardware optimizatio
Complex implementation, no room for hardware optimizatio	Burden on software to combine simpler operations to implement high-level language statements

ISA - Data Storage

Example for C = A+B

Stack	Accumulator	Register (load-store)	Memory-Memory
Push A	Load A	Load R1, A	Add C, A, B
Push B	Add B	Load R2, B	
Add	Store C	Add R3, R1, R2	
Pop C		Store R3, C	

ISA - Memory Addressing Mode

Big-Endian	Little-Endian
MSB stored in lowest address	LSB stored in lowest address

MIPS is Big-Endian

ISA - Operations in Instructions Set

Frequently Used Instructions

Rank	Instruction	Average %
1	Load	22%
2	Conditional Branch	20%
3	Compare	16%
4	Store	12%

Amdahl’s law – make the common cases fast!
If the total time originally:
T = (0.7 * t) + (0.3 * t)
Decrease common by 50%, uncommon by 50%
T = (0.7 * 0.5 * t) + (0.3 * 1.5 * t)
T = (0.80 * t) -> Faster

Datapath

Instruction Execution Cycle

1. Fetch

Use the PC to fetch instn from memory

Increment PC by 4

2. Decode

Read the `opcode` to determine instruction type

Read from all necessary registers

3. Execute

Performs	- Arithmetic, shifting, logical
	- Address calculation
	- Register comparison

Datapath (cont)

- Target address calculation

4. Memory

Use memory address calculated by ALU Stage

Read from or write to data memory

5. Register Write

Write into registers

Control Path

Control Signal	Execution	Purpose
`RegDst`	Decode/Fetch	0: Inst[20:16] 1: Inst[15:11]
`RegWrite`	Decode/Fetch /RegWrite	0: Idle 1: Register write
`ALUSrc`	ALU	0: Register RD 2 1: SignExt(Inst[15:0])
`ALUcontrol`	ALU	Select the operation to be performed
`MemRead /MemWrite`	Memory	0: Idle 1: Performs
`MemToReg`	RegWrite	1: Memory RD 0: ALU result
`PCSrc`	Memory /RegWrite	0: PC + 4 1: SignExt(Inst[15:0]) << 2 + (PC + 4)

ISA - Instruction

Fixed Length Instruction Encoding

Type-A	2 * 5-bit operands
Type-B	1 * 5-bit operand
Maximum	Maximise Type-B
	$1 + (2^6 - 1) * 2^5$
Minimum	Maximise Type-A
	$2^6 - 1 + 2^5$