INSTRUCTION SET ARCHITECTURE

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

THE UNIVERSITY OF UTAH

CS/ECE 6810: Computer Architecture

What is ISA?

- Instruction Set Architecture
  - Well-defined interfacing contract between hardware and software
  - Does define
    - The functional operations of units
    - How to use each functional unit
  - Does not define
    - How functional units are implemented
    - Execution time of operations
    - Energy consumption of operations

Example Problem

- Which one may be guaranteed by a ISA?
  - The number of instructions supported by processor
  - The number of multipliers used by processor
  - The width of operands
  - Sequence of instructions that results in an error
  - Sequence of instructions that results in lower energy consumption
  - The total number of instructions for an application program
  - The total amount of main memory (e.g., DRAM)

Example Problem

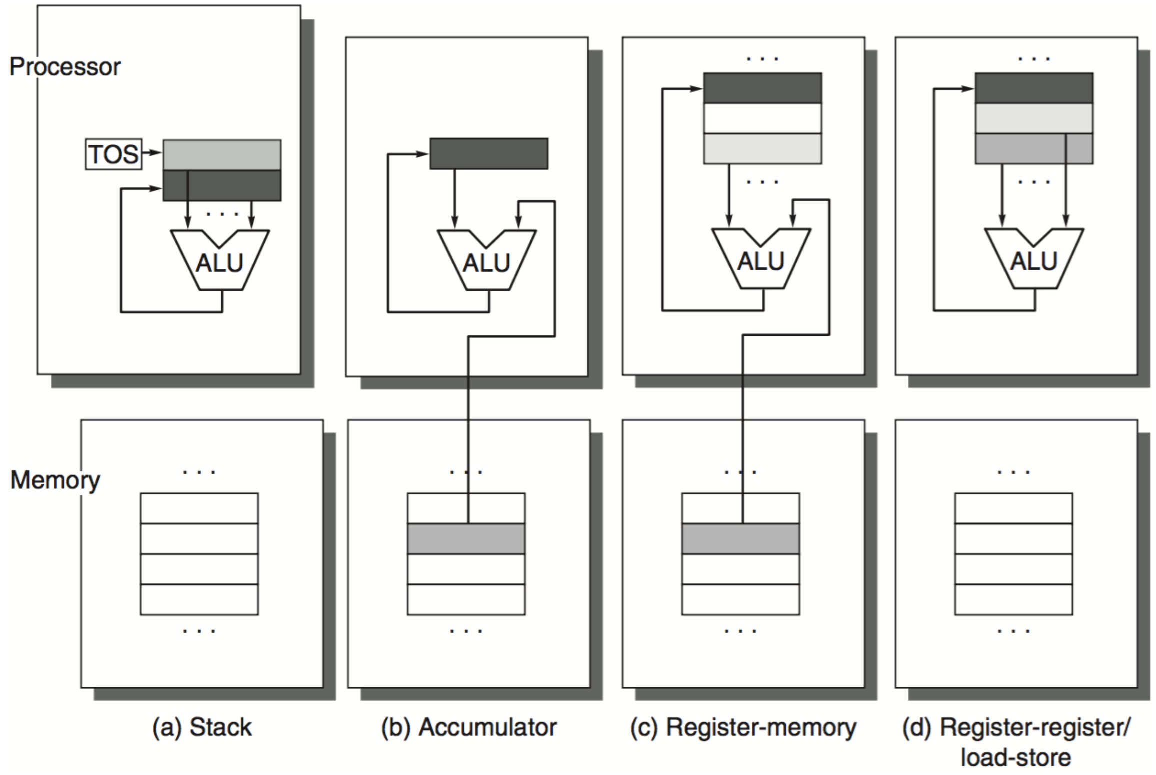
- Which one may be guaranteed by a ISA?
  - YES The number of instructions supported by processor
  - NO The number of multipliers used by processor
  - YES The width of operands
  - YES Sequence of instructions that results in an error
  - NO Sequence of instructions that results in lower energy consumption
  - NO The total number of instructions for an application program
  - NO The total amount of main memory (e.g., DRAM)

ISA to Programmer Interface

- Internal machine states
  - Architectural registers, control registers, program counter
  - Memory and page table
- Operations
  - Integer and floating-point operations
  - Control flow and interrupts
- Addressing modes
  - Immediate, register-based, and memory-based

ISA Types

- Operand locations



Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R3,R1,B	Load R2,B
Add	Store C	Store R3,C	Add R3,R1,R2
Pop C			Store R3,C

Which Set of Instructions?

- ISA influences the execution time
  - CPU time = IC x CPI x CT
- Complex Instruction Set Computing (CISC)
- Reduced Instruction Set Computing (RISC)

Which Set of Instructions?

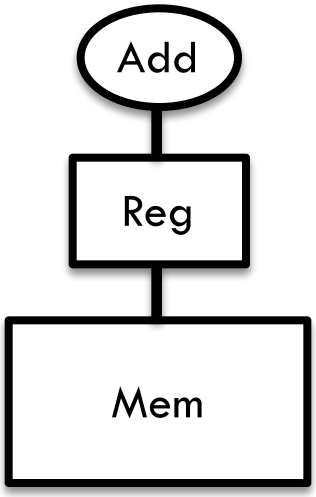
- ISA influences the execution time
  - CPU time = IC x CPI x CT
- Complex Instruction Set Computing (CISC)
  - May reduce IC, increase CPI, and increase CT
  - CPU time may be increased
- Reduced Instruction Set Computing (RISC)
  - May increases IC, reduce CPI, and reduce CT
  - CPU time may be improved

RISC vs. SISC

RISC ISA	CISC ISA
Simple operations	Complex operations
▪ Simple and fast FU	▪ Costly memory access
Fixed length	Variable length
▪ Simple decoder	▪ Complex decoder
Limited inst. formats	Limited registers
▪ Easy code generation	▪ Hard code generation

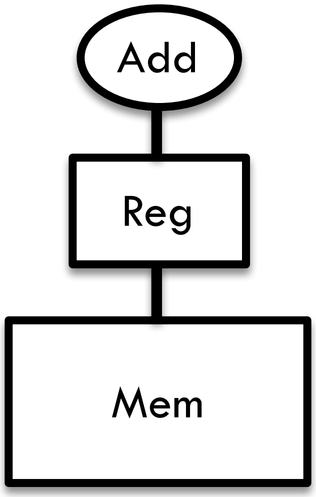
Memory Addressing

- Register
  - Add r4, r3
- Immediate
  - Add r4, #3
- Displacement
  - Add r4,100(r1)
- Register indirect
  - Add r4, (r1)



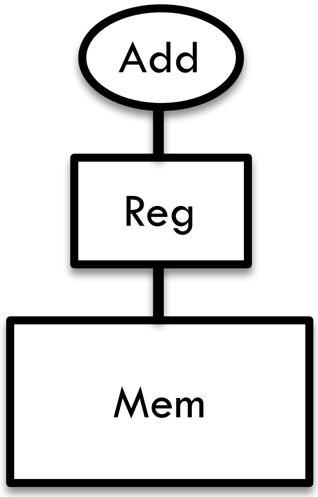
Memory Addressing

- Register
  - Add r4, r3 Reg[4]=Reg[4]+Reg[3]
- Immediate
  - Add r4, #3 Reg[4]=Reg[4]+3
- Displacement
  - Add r4,100(r1) ...+Mem[100+Reg[1]]
- Register indirect
  - Add r4, (r1) ...+Mem[Reg[1]]



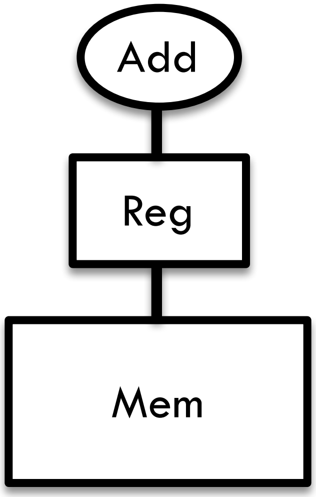
Memory Addressing

- Indexed
  - Add r3, (r1+r2)
- Direct
  - Add r1, (1001)
- Memory indirect
  - Add r1,@(r3)
- Auto-increment
  - Add r1, (r2)+



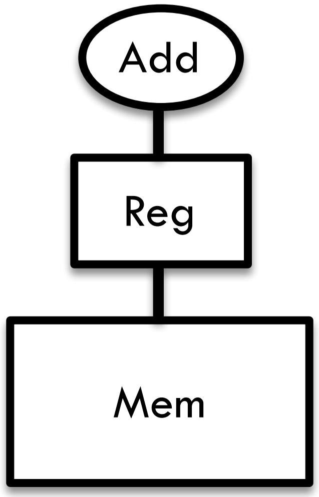
Memory Addressing

- Indexed
  - Add r3, (r1+r2)…+Mem[Reg[1]+Reg[2]]
- Direct
  - Add r1, (1001) …+Mem[1001]
- Memory indirect
  - Add r1,@(r3) …+Mem[Mem[Reg[3]]]
- Auto-increment
  - Add r1, (r2)+ …+Mem[Reg[2]]
  - Reg[2]=Reg[2]+d



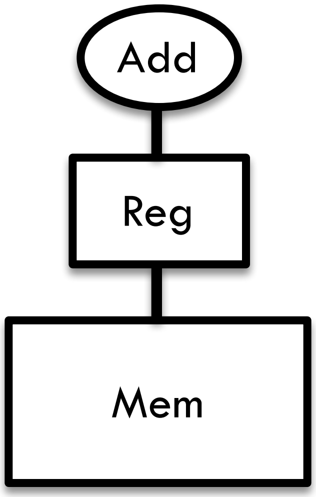
Memory Addressing

- Auto-decrement
  - Add r1, -(r2)
- Scaled
  - Add r1, 100(r2)[r3]



Memory Addressing

- Auto-decrement
  - Add r1, -(r2) Reg[2]=Reg[2]-d
  - …+Mem[Reg[2]]
- Scaled
  - Add r1, 100(r2)[r3]
  - …+Mem[100+Reg[2]+Reg[3] x d]



Example Problem

- Find the effective memory address
  - Add r2, 200(r1)
  - Add r2, (r1)
  - Add r2, @(r1)

Registers

r1	100
r2	200

Memory

…	…
100	400
200	500
300	600
400	700
500	800

Example Problem

- Find the effective memory address
  - Add r2, 200(r1)
    - r2 = r2 + Mem[300]
  - Add r2, (r1)
    - r2 = r2 + Mem[100]
  - Add r2, @(r1)
    - r2 = r2 + Mem[400]

Registers

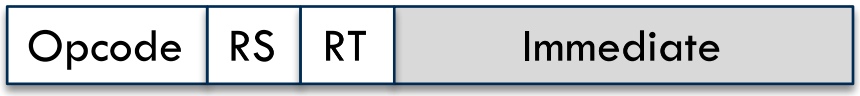
r1	100
r2	200

Memory

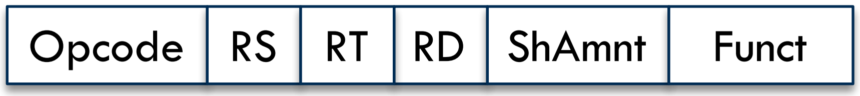
…	…
100	400
200	500
300	600
400	700
500	800

Instruction Format

- A guideline for generating/interpreting instructions
- Example: MIPS
  - Fixed size 32-bit instructions
  - Three opcode types
    - I-type: load, store, conditional branch



    - R-type: ALU operations



    - J-type: jump