Instruction Set Architecture (ISA) - Part I

CS6133 - Computer Architecture I

Vikram Padman

NYU Polytechnic School of Engineering

vikram@poly.edu

Agenda

1. **Introduction** - Part I
2. **Classifying ISA** - Part I, II, III and IV
3. **Simple CPU's ISA** - Part V

Reading List

- "Computer Architecture - A Quantitative Approach" - Appendix A in Fifth Edition or Appendix B in Fourth Edition
- "Computer Organization and Design" - Chapter 2 in Fourth Edition or Third Edition
- "Digital Design and Computer Architecture" - Chapter 6

Instruction Set Architecture

- In last lecture we saw modules that are used in the implementation of the simple MIPS CPU
- We went through a lot of details, but something was lacking

Instruction Set Architecture

- In last lecture we saw modules that are used in the implementation of the simple MIPS CPU
- We went through a lot of details, but something was lacking
- At this point you should be wondering:
  - Why does the control unit generate the control signal? and who defined those controls signal to begin with?
  - What could the simple CPU do?
  - There are a lot of MUXs in the data path, why? and again who defined them?
  - Don't really understand the rational behind the design

Instruction Set Architecture

- In last lecture we saw modules that are used in the implementation of the simple MIPS CPU
- We went through a lot of details, but something was lacking
- At this point you should be wondering:
  - Why does the control unit generate the control signal? and who defined those controls signal to begin with?
  - What could the simple CPU do?
  - There are a lot of MUXs in the data path, why? and again who defined them?
  - Don't really understand the rational behind the design
- All of the above valid questions.
- Before we could attempt to answer any of the above, we should look at the simple CPU's requirements.

Instruction Set Architecture

- Functional requirements and specifications for a CPU
- Co-Developed with software and evolves over time
- An ISA is:
  - Is the dictionary to a processor's assembly language
  - Defines a processor's capabilities and limitations
  - Specifics how the processor's capabilities could be accessed
  - Is the portion of the processor that is visible to a compiler or developers

Classifying ISA

An ISA's could be classified using the following:

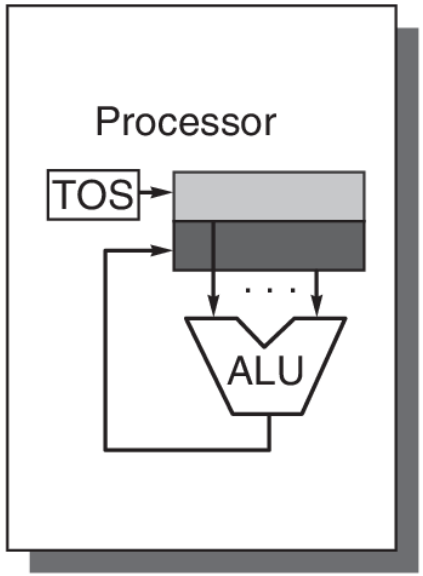
- Internal Storage - Part I
- Memory Addressing - Part II
- Type and Size of operands - Part II
- Operations in the instruction set - Part III
- Instruction for control flow - Part III
- Encoding an Instruction Set - Part IV

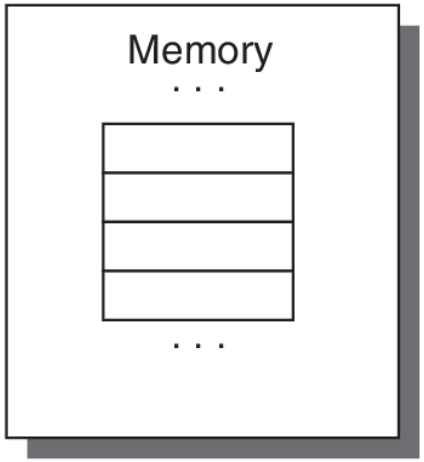
Internal Storage

Internal storage is one of the most basic architecture differentiator. There are essentially four classes or type of internal storage:

1. Stack
2. Accumulator
3. Register-Memory
4. Register-Register

Stack





- ALU only uses operands that are in the stack
- Top of the stack (TOS) is combined with the next value in the stack, TOS is removed and the result is stored in place of the second operand.
- Example instructions would be:
  - Push 10
  - Push 20
  - Add
  - Pop (30 is popped out)

Stack - Example Application DC

DC Commands:

- **Number** Push *Number* into stack
- **f** Print contents of the stack
- **Operation** Perform *Operation* on top two element
- **p** Pop top of the stack
- **q** Quit

```
vikram@KKV-SW ~ $ dc
10
20
30
40
50
f
50
40
30
20
10
+
f
90
30
20
10
+
f
120
20
10
+
f
140
10
+
f
150
p
150
q
vikram@KKV-SW ~ $
```

Stack - Advantages and Disadvantages

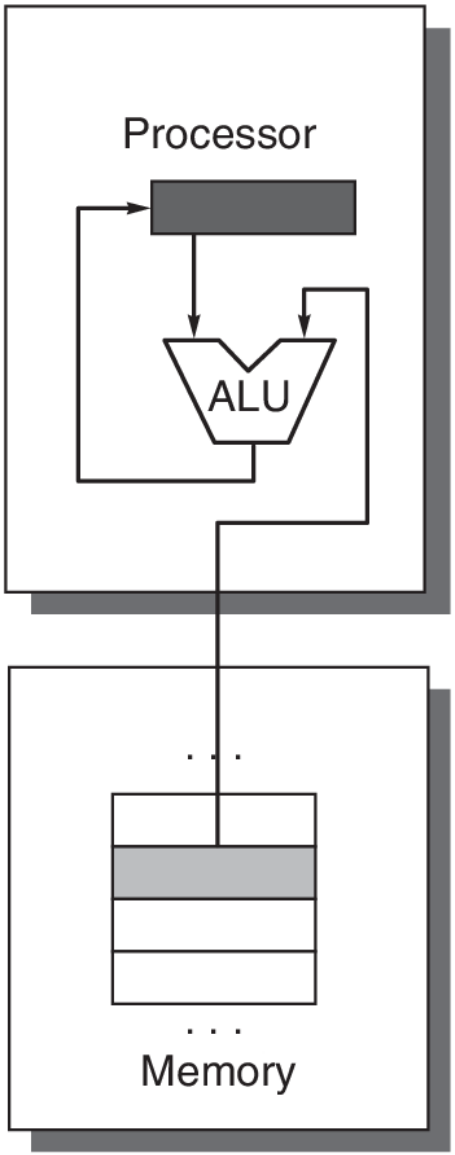
Advantages

- Compact instructions
- Simple Compiler / Interpreters
- Minimal processor states
- Much simpler and smaller hardware implementation

Disadvantages

- Increased memory usage
- Hard to factor out sub expressions or reuse previously calculated values
- Strict code order
- Much slower compared to other types

Accumulator



- ALU uses one operand from its internal storage and the other from the memory
- Result is stored back into operand 1's position.
- Example instructions would be:
  - Load 10
  - Add 20
  - Store <memory>

Accumulator - Examples

- Many early CPUs are accumulator machines, for example Intel 4004 and 8008
- Many embedded processor that in use today are accumulator machine. For example PIC processor and Atmel's AVR processor

Accumulator - Advantages and Disadvantages

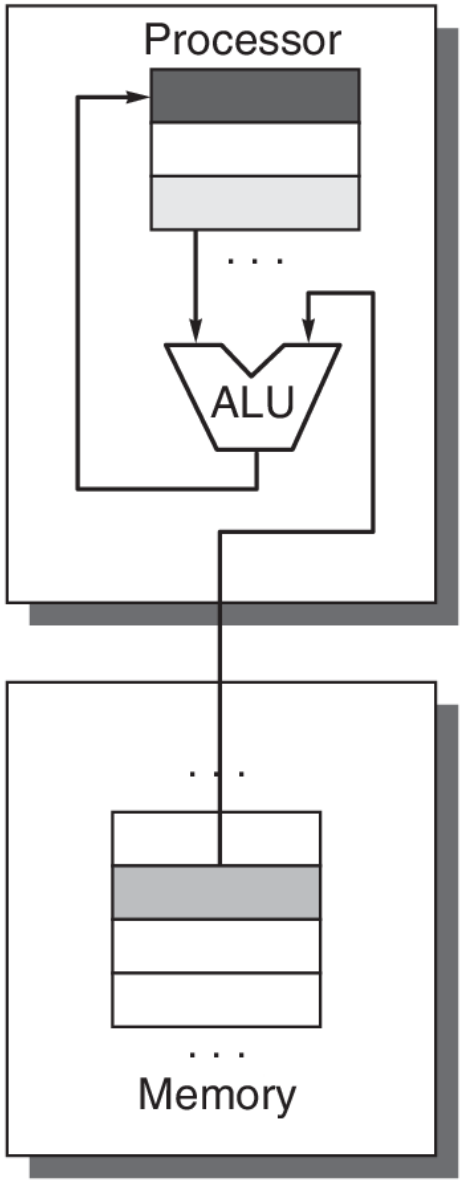
Advantages

- Compact instructions and reduced code density
- Simple Compiler / Interpreters
- Smaller hardware implementation

Disadvantages

- Increased memory usage
- Slower compared to other types

Register-Memory



- ALU uses one operand from its internal storage and the other from the memory
- Results are stored in a separate register
- Load-Store instructions are required to move data
- Example instructions would be:
  - Load R1, <memory>
  - Add R3, R1, <memory>
  - Store R3, <memory>

Register-Memory - Examples

- Modern x86 and x86-64 CPU's use register-memory architecture
- Legacy once are 8088, 386 and 486 families, IBM 360, Motorola 68K processor.

Register-Memory - Advantages and Disadvantages

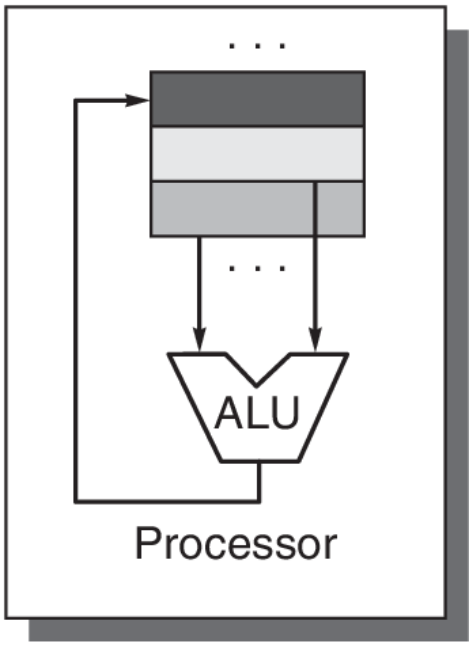
Advantages

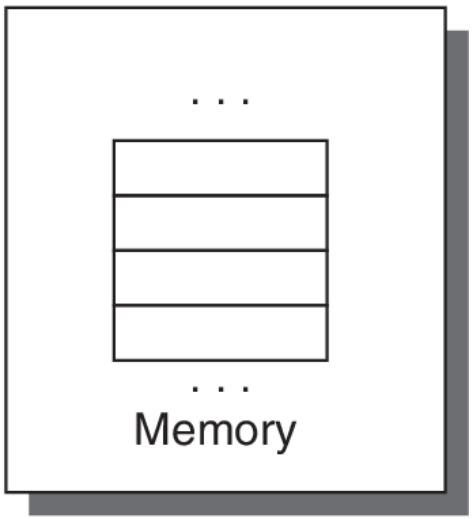
- A load could be eliminated.
- Easy to encode and decode instructions
- Yields good code density

Disadvantages

- Clock cycles to complete an instruction may vary by operand's memory location.
- Encoding memory address in each instruction may restrict the number of registers
- In a binary operation a source operand would be destroyed. For example: AND R1, R2

Register-Register





- All operands are in registers
- Results are stored in registers
- Load-Store instructions are required to move data
- Example instructions would be:
  - Load R1, <memory>
  - Load R2, <memory>
  - Add R3, R1, R2
  - Store R3, <memory>

Register-Register - Examples

- Modern ARM, SPARC, MIPS, PowerPC
- Legacy once are DEC Alpha, PA-RISC, SuperH

Register-Register - Advantages and Disadvantages

Advantages

- Simple, fixed length instructions
- Simple code generation model
- Similar number of clock cycles per instructions.

Disadvantages

- Higher instruction count
- Lower code density and larger executable

Week 5 - Activity 1

1. Give two examples of legacy stack based machines that was built and successfully used for commercial purpose. Provide architectural details that made them unique and practical.
2. What type of internal storage does Dr. Neumann's digital computer use? Justify your answer with details.

Week 5 - Activity 1

3. Read section A.1 (5th ED.) or B.1 (4th ED.) in "Computer Architecture - A Quantitative Approach" and answer the following questions:
   1. How are variables, used in a program, stored in a CPU?
   2. Is there an optimal number of registers a CPU should have? How would one calculate the optimal number of registers?
   3. What are the two types of CPUs with general purpose registers that are popular today? What type of internal storage do they use?
   4. Does compilers play a role in a CPU's efficiency? if yes, how? Justify your answer with details