Hardware Instructions - I

S.Venkatesan
Network Security and Cryptography Lab
Department of Information Technology
Indian Institute of Information Technology, Allahabad
venkat@iiita.ac.in

Acknowledgement: The contents and figures are copied from various sources. Thanks to all authors and sources made those contents public and usable for educational purpose

Hardware Communication

- Communication Language is Instructions (instruction set)
- Instruction Sets
  - RISC vs CISC: This is the classic difference between ARM/MIPS and x86.
  - RISC stands for Reduced Instruction Set Computer and CISC is Complex ISC.
  - x86 is a CISC processor and both ARM/MIPS are RISC.
  - **ARM** (stylised in lowercase as **arm**, formerly an acronym for **Advanced RISC Machines** and originally **Acorn RISC Machine**)
  - **MIPS** (**Microprocessor without Interlocked Pipelined Stages**) is a family of reduced instruction set computer (RISC) instruction set architectures (ISA)

Operations and Operands

ADD A, B, C

ADD is the operation

A, B and **C** are operands

ARM

- 32 bit 16 registers; smaller is faster
- 13 registers for user operands and Stack Pointer, Link Register, Program Counter are for system.
- ARM v6/v7 maintains a status register called the **CPSR** (current program status register) that holds four status bits, negative (N), zero (Z), carry (C), and overflow (O). These bits can be used for conditional execution of subsequent instructions.
- Groups of 32 bits called word
- In ARM, words must start at address that are multiples of 4. [Alignment restriction]
- Little endian and Big endian
- ARM is little-endian

Little endian and Big endian

- In little-endian format, the byte with the lowest address in a word is the least-significant byte of the word. The byte with the highest address in a word is the most significant. The byte at address 0 of the memory system connects to data lines 7-0.

- In big-endian format, the byte with the lowest address in a word is the most significant byte of the word. The byte with the highest address in a word is the least significant. The byte at address 0 of the memory system connects to data lines 31-24.

ARM Assembly Language Instructions

- Arithmetic – add (ADD), subtract (SUB)
- Data Transfer – load (LDR), store(STR), move (MOV), swap (SWP), etc,
- Logical – and (AND), or (ORR), not (MVN) , logical shift left (LSL) and right (LSR)
- Conditional Branch – compare (CMP), Branch or EQ, NE, .. (BEQ 25)
- Unconditional Branch – branch (always) (B 2500), branch and link (BL 2500)

Operations/Instructions

ADD a, b, c

Rigid: One operation and three variables.

How can we add more values – by doing the following

ADD a, b, c ; sum of b and c to a
ADD a, a, d ; sum of b,c,d to a
ADD a, a, e ; sum of b,c,d,e to a

Data Transfer Instructions

- Transfer data between memory and registers
- To access a word in memory, memory address to be supplied in the instruction.
- Storage is an array that is in sequence.

  g = h + A[8]

- Uses **Load** to transfer data from memory to register

  LDR

  Syntax: **LDR r5,[r3,#32]**

  ADD r1, r2, r5 ; where r2 have h and r5 have A[8]

Complement to Transfer

STR r5, [r3,#48] ; stores h+A[8] to A[12]

Constant or Immediate Operands

- **LDR r5, [r1, #AddrConstant4]** ; r5 = constant 4
- **ADD r3, r3, r5** ; r3 = r3 + r5 (r5 == 4)

- **ADD r4, r3, #4** ; r3 = r3 + 4

Spilling registers

- To put the less commonly used variables.
- Arithmetic instruction – read two registers and operate.
- Data Transfer instruction – Reads or writes one operand without operating on it.

Von-Neuman Architecture

- Historically there have been 2 types of Computers:
  - **Fixed Program Computers –** Their function is very specific and they couldn't be programmed, e.g. Calculators.
  - **Stored Program Computers –** These can be programmed to carry out many different tasks, applications are stored on them, hence the name. [The modern computers are based on a stored-program concept introduced by John Von Neumann.]

```
.cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    movl    $10, -4(%rbp)
    movl    $12, -8(%rbp)
    movl    -4(%rbp), %eax
    addl    %eax, -8(%rbp)
    movl    $0, %eax
    popq    %rbp
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
```

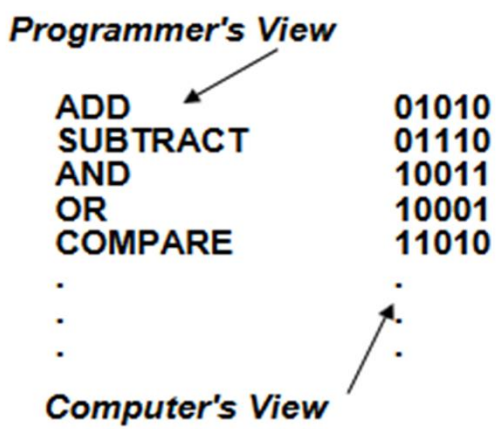
```
#include<stdio.h>
int main() {
        int b, c;
        c = 10l;
        b = 12;
        b = b + c;
        return 0;
}
```

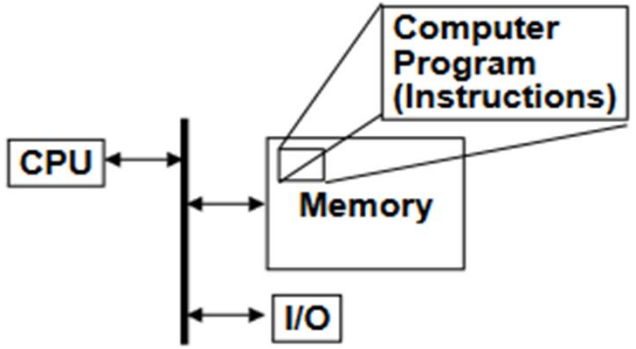
x64

```
.cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    movl    $10, -4(%rbp)
    movl    $12, -8(%rbp)
    movl    $113, -8(%rbp)
    movl    $0, %eax
    popq    %rbp
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
```

```
#include<stdio.h>
int main() {
        int b, c;
        c = 10l;
        b = 12;
        b = 12 + 101;
        return 0;
}
```

Instruction Set Architecture





Princeton (Von Neumann) Architecture

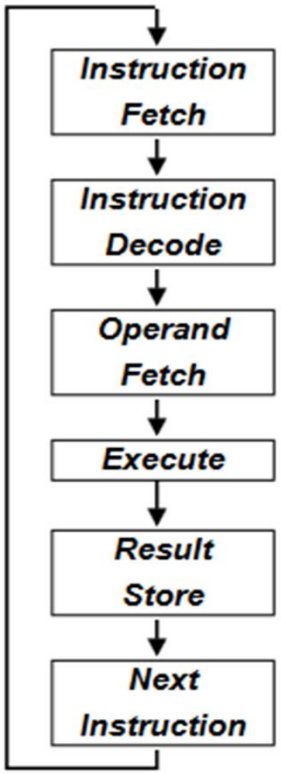
- --- Data and Instructions mixed in same memory ("stored program computer")
- --- Program as data (dubious advantage)
- --- Storage utilization
- --- Single memory interface

Harvard Architecture

- --- Data & Instructions in separate memories
- --- Has advantages in certain high performance implementations

Thanks to Dr. Nabil J. Sarhan, Wayne State University

Execution Cycle



Obtain instruction from program storage

Determine required actions and instruction size

Locate and obtain operand data

Compute result value or status

Deposit results in storage for later use

Determine successor instruction

Thanks to Dr. Nabil J. Sarhan, Wayne State University

Thank You