Lecture 4-5: February 19, 2021
Computer Architecture and Organization-I
Biplab K Sikdar

0.0.1 Extension of basic organization

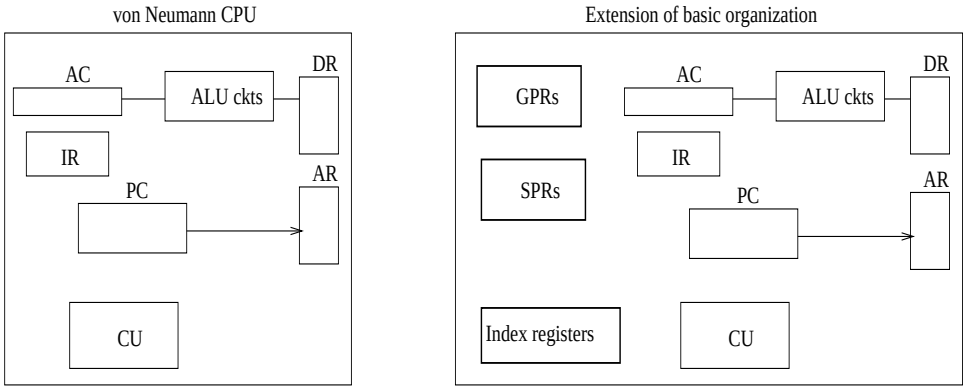

Figure 1: Extension of basic von Neumann architecture

The basic von Neumann architecture is made more powerful by several ways.

1. Additional registers are added. These are:

- General purpose (multi purpose) registers. This can be viewed as the replacement of single accumulator by a set of registers. These are explicitly addressable by the processor through instructions.
- Index registers. While defining operands in an instruction the instruction may also specify the index address.

  The content of index register is then added to the operand address specified in instruction to get the (effective) address of operand.
- Special purpose registers - SP, segment registers, status register etc.

2. Capabilities of the ALU are enhanced. In basic von Neumann architecture, simple arithmetic was introduced. In the modern design,

- The multiplication, division and other complex instructions are considered.
- Floating point hardware is introduced.

3. Instruction prefetching is introduced. Consider execution of I, I+1, I+2, $\cdots$. Concept of basic von Neumann architecture is- fetch I from memory and execute. I+1 is fetched only after completion of I. In instruction prefetching, I+1, I+2, $\cdots$, that are otherwise ready for execution, are fetched while I is in execution (Fig. 2).

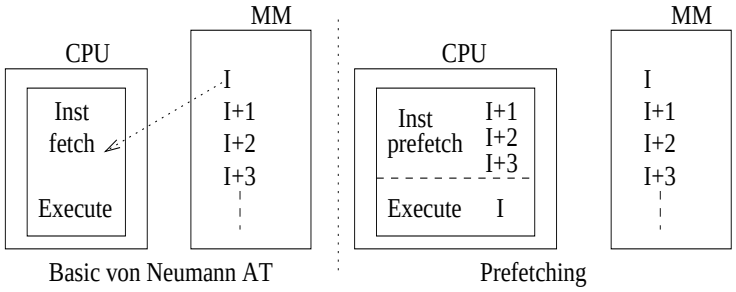

Figure 2: Instruction prefetching

4. Special control circuitry added: Initial proposal was a single CU(). At present, CPU can have multiple control units. For example, in addition to main CU (called supervisor CU) CPU is having multiplier CU (the slave unit, Figure 3).

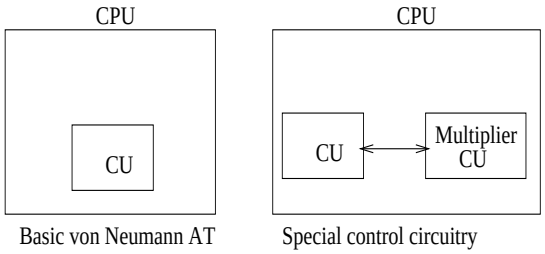

Figure 3: Special control circuitry

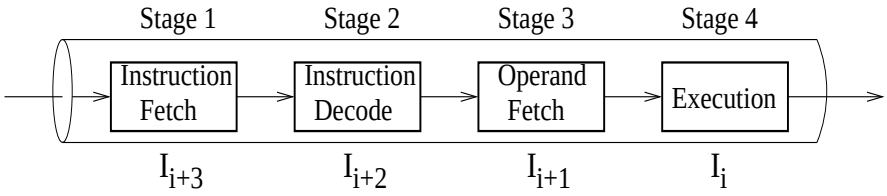

Figure 4: Pipelining

5. Speed up through pipelining within the CPU (Figure 4).

6. Parallel processing: Simultaneous processing of two or more distinct instructions. Several instructions and/or operands are fetched simultaneously.

Execution of more than one instruction is overlapped. The options are

- ALU is divided into K-parts (Figure 5) - K partitioned ALU. It allows simultaneous execution of one integer instruction, a floating point instruction, one logical operation etc.

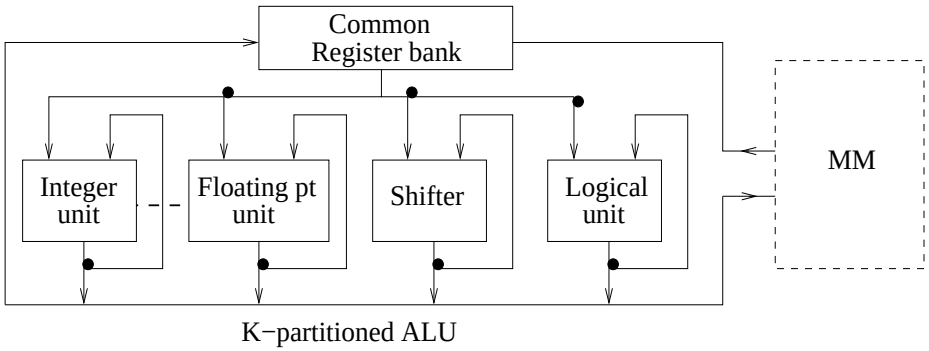

Figure 5: CPU with K-partitioned ALU

- ALU is replicated K-times (Figure 6). The K copies of an ALU circuit allow simultaneous execution of K instructions.

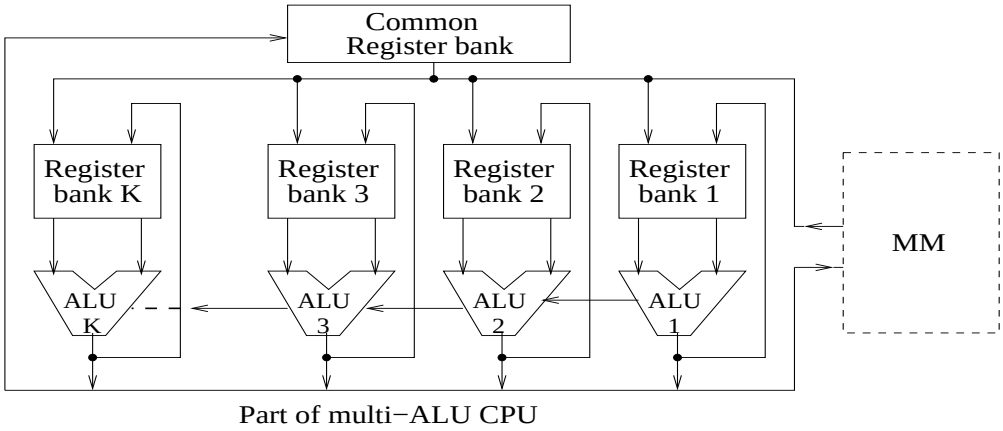

Figure 6: CPU with replicated ALUs

Latest introductions in parallel processing architecture are - multiple processor core, very large instruction word (VLIW) architecture, superscalar architecture etc.

7. Speculative execution:

8. Green computing:

Instruction Set Architecture

The main function of the CPU is to execute user program stored in memory.

CPU can only directly fetch instruction of a program from MM.

The instruction is then decoded, operands of instruction (if there is any) is fetched, and finally the execution of instruction is done by CPU.

CPU can execute only valid instructions (called macro instruction) specified in its instruction set.

Instruction set architecture of a CPU refers to programmer visible instruction set.

The type of instructions that should be included in a general purpose processor's intruction set can not be arbitrary.

The insruction set architecture design follows a few guidelines.

0.1 Instruction Set Design

Following requirements are to be fullfilled while defining instruction set of a m/c.

1. *Complete*: It means - a programmer should be able to construct machine language program to solve most of the common problems.

2. *Efficient*: Instruction set of a computer is to be such that - instructions used most frequently by the programmers take less time to execute.

   That is, instructions like simple data movement, integer arithmatic instructions, simple logical instructions, branch control instructions etc. should take less time for execution.

3. *Regular*: If an instruction I_T performs task T, there should be an instruction I_U that can undo the task T.

   For example, if an instruction I_{Lshift} is included to perform one bit left shift of accumulator, then there must be an I_{Rshift} instruction that can perform one bit right shift on accumulator.

   Similarly, if INR (increment content of a register) is considered, then to be regular DCR must be included.

4. *Compatible*: To reduce hardware and software costs, an instruction set must be compatible with existing machine instructions.

It ensures - design cost for implementing the instructions becomes affordable.

Example: Say, the design for 16-bit addition logic is available. Then inclusion of 32-bit addition instruction can be affordable. However, the choice of 24-bit addition instruction may result in exorbitant implementaion cost.

Generic instructions types conventionally included in a computer instruction set

i) Data transfer (data movement) instructions: Example - Move, Load, Store, Swap, Push, Pop, Clear, Set etc.

ii) Arithmatic instructions: Example- Add, Sub, Multiply, Division, Increment, Decrement, Negate, Absolute etc.

iii) Logical instructions: Example- Set, Reset, Rotate, And, Or, Ex-or, Not, Shift, Translate, Convert, Edit etc.

iv) Program control instructions: Example- Execute, Skip, Jump, Test, Compare, Set control variables, Wait, Nop, Call, Return, Halt etc.

v) Input/output instructions: Example- Read, Write, Start I/O, Halt I/O, Test I/O or channel etc.

vi) Special purpose instructions: Example- Diagnose, Trap, Breakpoint, instructions for operating system etc.

0.2 Instruction Format

Follow Figure 7.

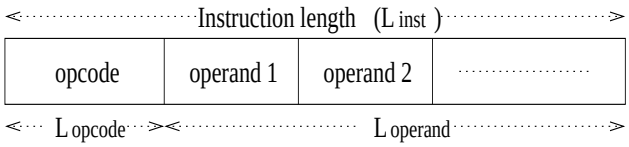

Figure 7: Instruction format

There can be zero or more operands - explicitly specified.

L_{inst} is the instruction length (L_{opcode} + $L_{operand}$).

Memory requirement to store a program depends on the length of individual instruction as well as the number instructions needed.

A main memory capable of delevering N bits/s can make N/L_{inst} number of instructions/s available to the CPU.

A CPU is having shorter instruction length can access larger number of instructions per cycle leading to a faster program execution.

Further, instruction length has the effect on size of instruction decoder and the CPU registers such as DR, IR etc.

Normally, widths of CPU data paths are same as that of word size/half-word size/byte. Therefore, CPU instruction length of one word/half-word/byte is desirable.

Optimization of instruction length follows two options

a) Optimization of the length of operand field, and

b) Optimization of the opcode length.

0.3 Operands Optimization

Length of operand field is optimized by reducing the explicit operands as well as through optimization of the legth of individual operand field.

Example: in a processor if there are 32 general purpose registers, the instruction

$$\text{ADD } R_i \text{ (AC} \leftarrow \text{AC + } R_i)$$

requires 5 bits to specify R_i mentioned explicitly. On the other hand, 10 bits are required for specifying operands of

$$\text{ADD } R_i, R_j \text{ (} R_i \leftarrow R_i + R_j)$$

Here, the two operands R_i and R_j are mentioned explicitly.

0.3.1 m-addressed machine

Number of operands explicitly specified in an instruction is reduced.

For an operation with n operands, only m of them are explicitly mentioned in the instruction format, where $m \leq n$. The rest n-m operands are implicit.

For example, ADD X (AC$\leftarrow$ AC + M(X)) specifies only one operand (X) explicitly.

However, implementation of such an instruction requires 3 operands.

The implicit other two operands are the AC (accunulator).

Conventionally instruction set are designed considering m = 0/1/2/3.

Small m signifies reduced size of an instruction.

However, if number of explicit operands (m) is reduced, the number of instructions needed to get solution to a problem increases.

Therefore, choice of an optimum m is to be decided.

Definition 0.1 *A computer is called an m-addressed machine if each of the instructions in its instruction set is p-address, where $p \leq m$ and m =1, 2, $\cdots$.*

Definition 0.2 *In a zero-addressed machine (m = 0), all the instructions except PUSH and POP are 0-address.*

0.3.1.1 3-address instruction

Here, number of explicit operands in an instruction is m=3. Instruction of the form

$$\text{ADD A, B, C} \quad \Rightarrow \text{A}\leftarrow\text{B + C}$$

is an example of 3-addressed instruction.

0.3.1.2 2-address instruction

Example, in the follwing 2-addressed instruction

$$\text{ADD A, B} \quad \Rightarrow \text{A}\leftarrow\text{A + B,}$$

the A represents two operands - one explicitly specified and other one is implicit.

0.3.1.3 Single-address instruction

In one-address (or single-address) instruction, only one operand is explicitly specified in the instruction representation.

All other operands of the instruction are implicit.

In the single-address instruction

$$\text{ADD B} \quad \Rightarrow \text{AC}\leftarrow\text{AC + B,}$$

the other two operands are the accumulator (AC).

0.3.1.4 Zero-address instruction

In a zero-address instruction representation, all the operands are implicit. Example:

$$\text{ADD} \quad \Rightarrow \text{STACK[Top]}\leftarrow \text{STACK[Top] + STACK[Top-1].}$$

Two top elements of STACK are added and result is placed on the top of STACK.

The zero-addressed m/c may not be the best choice.

Though instruction length decreases, the program size increases interms of number of instructions required to write the solution to a problem.

Example 0.1 Consider evaluation of Z = w - x + y.

The solution in 3-addressed m/c requires 2 instructions.

$$\begin{aligned} \text{SUB } Z,\ w,\ x &\Rightarrow Z \leftarrow w - x \\ \text{ADD } Z,\ Z,\ y &\Rightarrow Z \leftarrow Z + y \end{aligned}$$

The solution in 2-addressed m/c requires 3 instructions.

$$\begin{aligned} \text{ASS } Z,\ w &\Rightarrow Z \leftarrow w \\ \text{SUB } Z,\ x &\Rightarrow Z \leftarrow Z - x \\ \text{ADD } Z,\ y &\Rightarrow Z \leftarrow Z + y \end{aligned}$$

In 1-addressed m/c, the number of instructions is 4.

$$\begin{aligned} \text{LOAD } w &\Rightarrow AC \leftarrow w \\ \text{SUB } x &\Rightarrow AC \leftarrow AC - x \\ \text{ADD } y &\Rightarrow AC \leftarrow AC + y \\ \text{STORE } Z &\Rightarrow Z \leftarrow AC \end{aligned}$$

The solution in 0-addressed m/c

```
PUSH  w
PUSH  x
SUB
PUSH  y
ADD
POP  Z
```

requires 6 instructions.