Computer Organization

Instruction Set Characteristics, Instruction Formats, Addressing Modes, RTL & Micro-Operations, CISC, RISC.

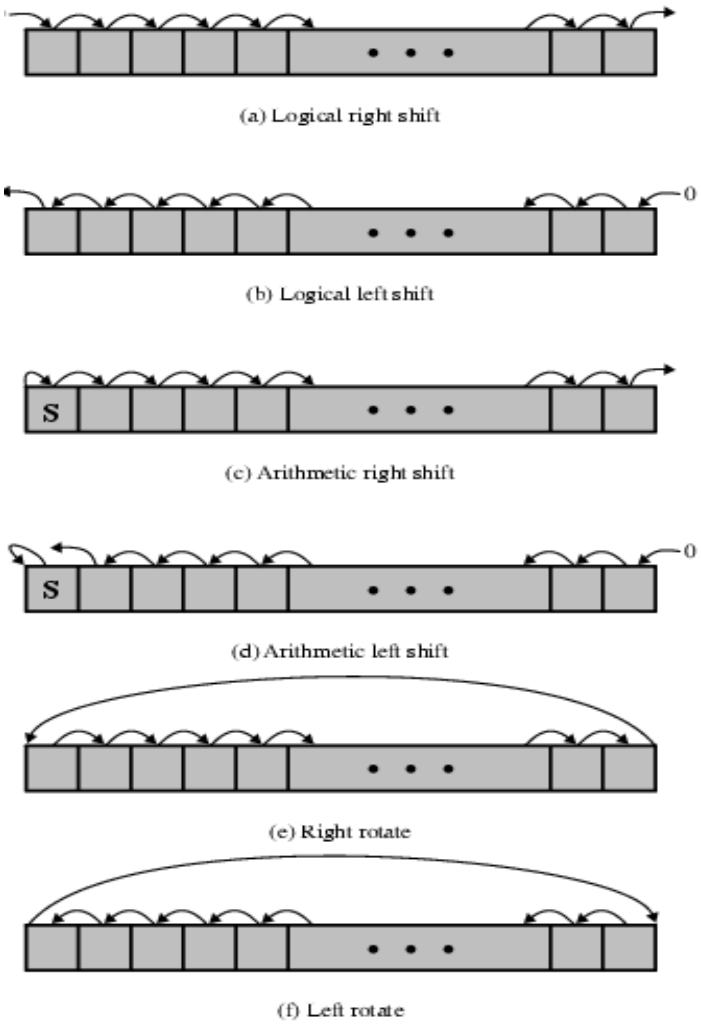
Chapters (10 + 11 + Mano Ch.4 + 13)

Typical Instructions

Data Movement	Load (from memory) Store (to memory) memory-to-memory move register-to-register move input (from I/O device) output (to I/O device) push, pop (to/from stack)
Arithmetic	Data Types: (signed & unsigned) Integer (binary + decimal) (signed & unsigned) Floating Point Numbers Operations: Add, Subtract, Multiply, Divide
Logical	Not, and, or, set, clear
Shift	Arithmetic (& Logical) shift (left/right), rotate (left/right)
Control (Jump/Branch)	unconditional, conditional
Subroutine Linkage	call, return
Interrupt	trap, return
Synchronisation	test & set (atomic r-m-w)
String	search, compare, translate

Types of Operation

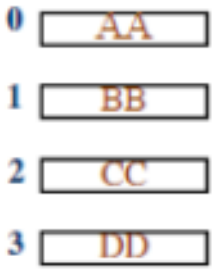
Shift and Rotate Operations

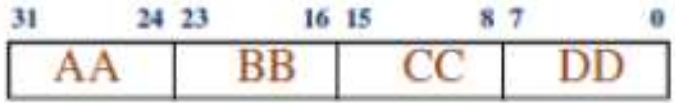


Types of Operand

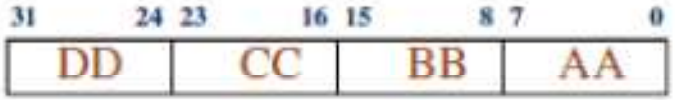
- Addresses
- Numbers
  - Integer/floating point
- Characters
  - ASCII etc.
- Logical Data
  - Bits or flags

Endianess





big endian



little endian

C to Assembly

- C code segment:

  a = b + (d >> 2);

  c = a*2 + 3/a

  Compiler => Assembly

  shr d, 2

  add a, b, d

  mul t0, a, 2

  div t1, 3, a

  add c, t0, t1

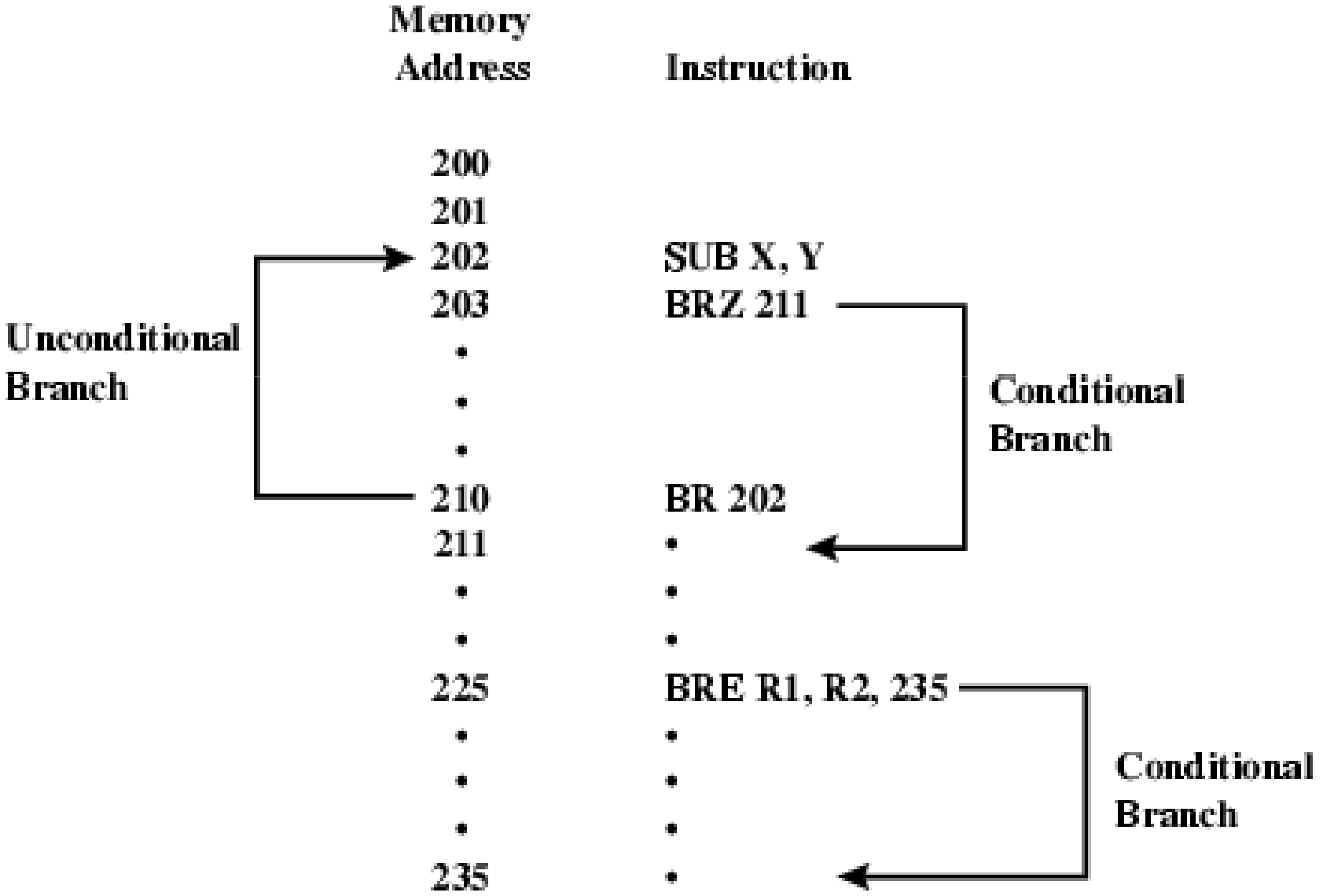
Transfer of Control

- Branch
  - e.g. ***BRZ X*** branch to x if result of (ADD,SUB,…) is zero
  - Uses condition bits register
  - See next slide
- Skip
  - e.g. increment and skip if zero ISZ

```
301
 .
 .
309      ISZ  R1
310      BR   301
311
```

  * eg. R1is set to -1000, the loop will be executed 1000 times
- Subroutine call
  - c.f. interrupt call

Branch Instruction



C to Assembly

- C code segment:

```
If( a == 5 )    b = a*2 + 3
Else            b = a*2 + a + 3
```

Compiler => Assembly

```
    BRE a, 5, IF
    mul t0, a, 2
    add t0, a
    add t0, 3
    BR Exit
IF:
     mul t0, a, 2
    add t0, 3
Exit:
```

C to Assembly

- C code segment:

```
while ( a >= 10 ) {
   b += (a>>1 )
   a -= 1
}
```

Compiler => Assembler

```
WL:
    sub t0, a, 10;
    BRN Exit
    shr t1, a, 1
    add b, t1
    sub a, 1
Exit:
```