Chapter 04: Instruction Sets and the Processor organizations

Lesson 11:

Generation of Memory Addresses and Addressing Modes

Objective

- Learn how a memory address generates in different addressing modes

Memory address generation

Operand in the instructions of instruction set

- An operand of an instruction is either at a register or is the immediate operand (part of instruction after the opcode bits or at a memory address

Addressing mode in an instruction set

- To compute the source or destination address of the operands

CISC

- Many addressing modes for computing the memory addresses

Need to compute the address of the operands

- Memory addresses when completely and directly addressed makes the instruction long

(i) Suppose 32-bits address is directly specified in the two operands of an instruction, the instruction will become more than 64 bits

Need to compute the address of the operands

(ii) Further, many a times, the address needs to be varied, for example, in processing data at a table

In the table, operand address computes from table base address, row number data and column number data

Addrerssing mode

Addressing Modes

- Earlier discussed four modes

(i) Register [Syntax ri, Address ri,]

(ii) Memory direct or absolute [Syntax M [$addr_i$], Address $addr_j$.]

(iii) Register indirect memory [Syntax r_i{M [$addr_j$]} or simply (r_i), Address $addr_j$ pointed by value in r_i] and

(iv) Immediate operand [Syntax #data_value, Operand data_value.]

Examples

(i) MOV r1, r4; ADD r1, r4;

(ii) LD r4, M [0x1300 0000];

- ST M [0x1300 0000], r0;
- ST M [0x1300 0004], r4

ST (r2), r1

- r1 is also a 32-bit register
- First the instruction I is fetched in a read cycle k from an address i after PC ← i
- Then in a write cycle k + 1, the register *r1* bits are transferred to addresses j to j +3, address j pointed by the register r2

Examples of modes

- *(iii) ST (r2), r1; where r1* is also a 32-bit register. First the instruction I is fetched in a read cycle k from an address i after PC ← i. Then in a write cycle k + 1, the register *r1* bits are transferred to addresses j to j +3, because address j pointed by the register r2
- (iv) LD r4, #0x100;

Fifth Other Addressing Mode

- (v) **Index** [Syntax *x*(*ri*), Address *ri* + *x*,] and [Syntax (*ri*), Address *ri*] when offset = 0

LD –0x80(r4)?

- Assume─ r4 is 0x13000
- –0x80 is the displacement defined at the instruction
- Address referenced by the instruction is 0x13000 – 0x80

Index addressing mode

- Label plus immediate addressing mode adds the value of the immediate to the value of the register to get the destination address
- Adding –0x80 to 0x13000 gives 0x12F80, the address referenced by during the load instruction
- r4 unchanged.

Sixth Other Addressing Mode

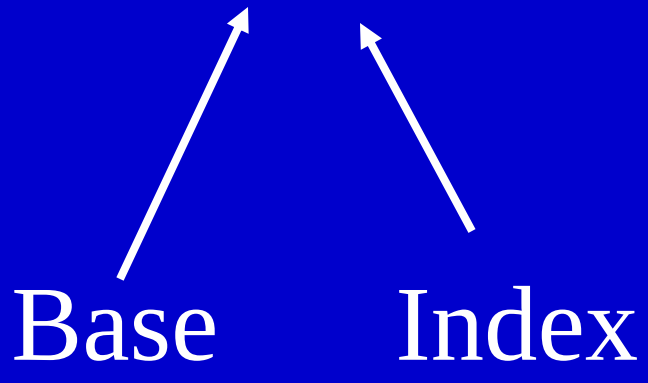
(vi) **Base-Index** [Syntax (*ri*, *rj*), Address *ri* + *rj*]

Seventh Other Addressing Mode

- (vii) **Base-Index Offset** [Syntax *x* (*ri*, *rj*), Address *ri* + *rj* + *x*]

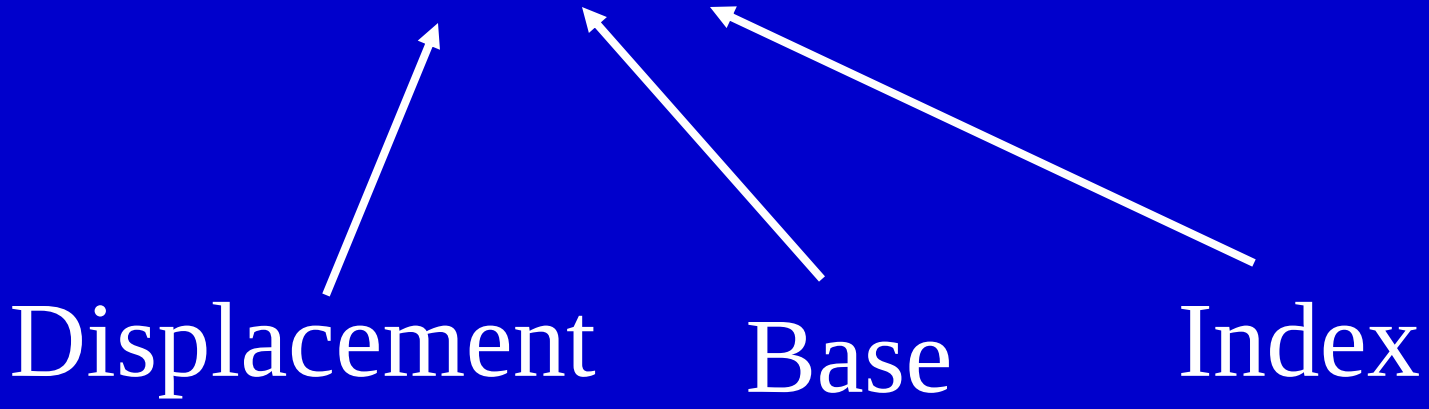
Example Base Index

- LD r0, (r8, r9)



Example Base-Index Relative

- LD r0, 32(r8, r9)



Eight Other Addressing Mode

- (viii) **Relative** addressing [Syntax *rel* or *x(PC)*, Address *PC* + *rel* or *PC* + *x*]

BR label, rel

- Assume: The linker calculates the offset from the branch instruction, label *rel* = 0x437 bytes

Target addresses

- Target address of the branch = Sum of the address of the branch (the PC when the branch executes) and the offset
- Assume the branch instruction loaded info at address 0x4000
- Target address is 0x4437 (0x4000 + 0x437)
- When the branch is loaded into address 0x4400, the target address is 0x4837.

Ninth Other Addressing Mode

- (ix) Auto-decrement or auto pre-decrement— first decrement a register, then use a register indirect [Syntax –(ri), Address ri $-1 \times$ L]
- L the length of the each memory word in bytes

Tenth Other Addressing Mode

(x) **Auto-increment** or auto post-increment use a register indirect and then increment a register for accessing next word at memory in later instruction [Syntax (*ri*)+, Address *ri* $+1 \times l$]

Auto post Increment

- Syntax (ri)+ — address is ri in present instruction, but will be *ri* +1× *l* in next time use of ri as ri post increments
- *l* = is the word length in bytes

Post Increment Offset Addressing

- Post increment offset addressing using *ST x[r2], r1* Instruction

Different architectures using different syntaxes

- Post-increment addressing an architect may have the separate instructions for post-incrementing and non post-incrementing addressing modes

Summary

We learnt

- Memory address generation in different addressing modes
- Base
- Base Relative
- Index
- Index Relative

We learnt

- Base-Index
- Base-Index Relative
- Auto Post Increment
- Auto Predecrement

End of Lesson 11 on

Generation of Memory Addresses and Addressing Modes