Basic Addressing Modes

Humayun Kabir
Professor, CS, Vancouver Island University, BC, Canada

Basic Addressing Modes: Outline

- Immediate
- Direct
- Indirect
- Register
- Register Indirect
- Displacement
- Stack

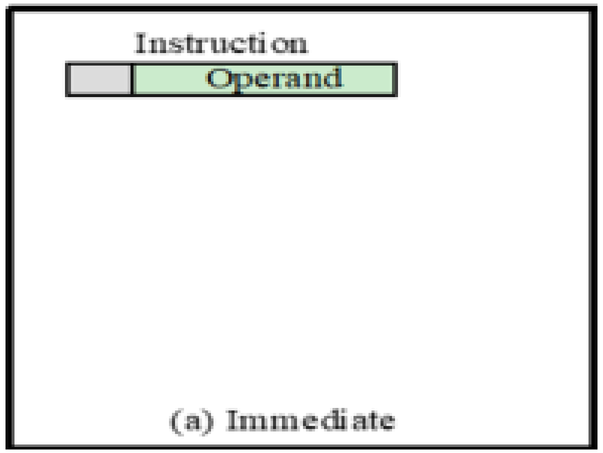
Basic Addressing Modes

Ways to refer the operands in assembly language instructions are called **addressing modes**.

```
mov x3, #1
cmp x3, x4
add x0, x1, x2
ldr x2, [x1]
add x3, x3, #1
```

Basic Addressing Modes: Immediate

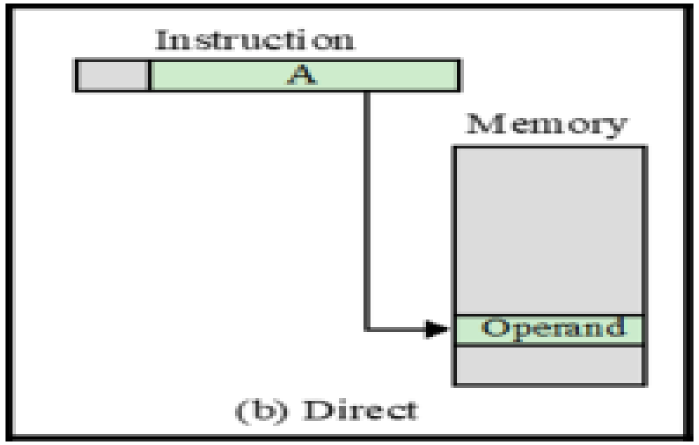
- Simplest form of addressing, address field contains the operand, e.g., mov x3, #1
- A = Address field, and Operand = A
- Advantage:
  - No memory reference other than the instruction fetch is required to obtain the operand, thus saving one memory or cache cycle in the instruction cycle
- Disadvantage:
  - The size of the number is restricted to the size of the address field, which, in most instruction sets, is small compared with the word length



Basic Addressing Modes: Direct

- Address field contains the effective address of the operand,

  str x0, [0x800c]
- A = Address field, Effective Address(EA) = A, and Operand = (A)
- Parentheses ( ) are to be interpreted as meaning *contents of.*
- Was common in earlier generations of computers.
- Requires only one memory reference and no special calculation
- Limitation is that it provides only a limited address space.

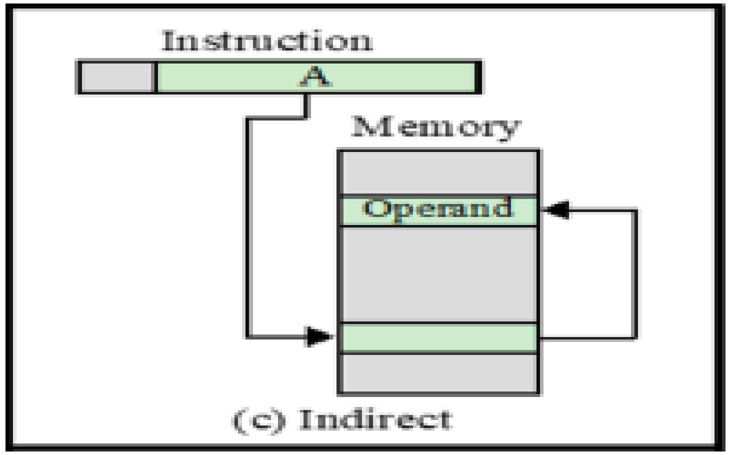


Basic Addressing Modes: Indirect

- Reference to the address of a word in memory which contains a full-length address of the operand.

  ldr x0, [[0x800c]]

- EA = (A), Operand = ((A))
- Advantage:
  - For a word length of *N* an address space of 2^N is now available
- Disadvantage:
  - Instruction execution requires two memory references to fetch the operand; one to get its address and a second to get its value



Basic Addressing Modes: Register

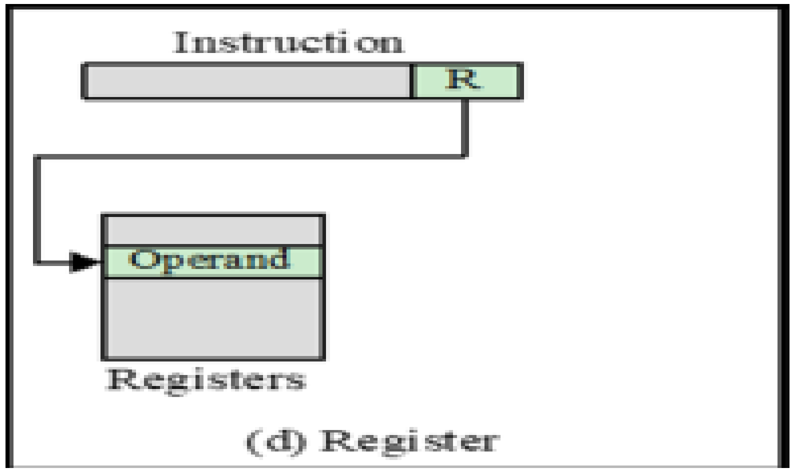
- Address field refers to a **register** rather than a main memory address, e.g., add x0, x1, x2
- EA = R, Operand = (R)

Advantages:

- Only a small address field is needed in the instruction
- No time-consuming memory references are required

Disadvantage:

- The address space is very limited

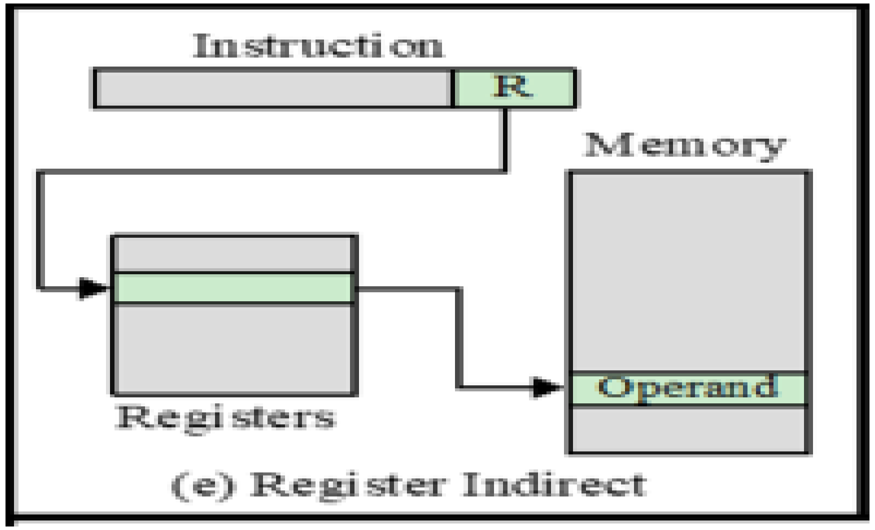


Basic Addressing Modes: Register Indirect

- Analogous to indirect addressing, the only difference is whether the address field refers to a memory location or a register, e.g.,

```
ldr x0, [x1]
```

- EA = (R), Operand = ((R))
- Address space limitation of the address field is overcome by having that field refer to a word-length location containing an address
- Uses one less memory reference than indirect addressing

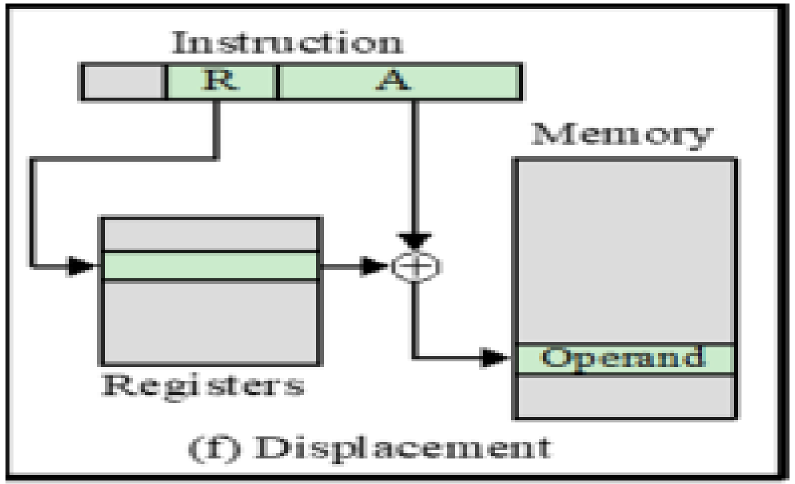


(e) Register Indirect

Basic Addressing Modes: Displacement

- Combines the capabilities of direct addressing and register indirect addressing, e.g., str x0, [x1, #16]
- EA = A + (R), Operand = (A+(R))
- Requires that the instruction have two address fields, at least one of which is explicit
  - The value contained in one address field (value = A) is used directly
  - The other address field refers to a register whose contents are added to A to produce the effective address

- Most common uses:
  - Relative addressing
  - Base-register addressing
  - Indexing



Basic Addressing Modes: Relative Addressing

The implicitly referenced register is the program counter (PC)

- The next instruction address is added to the address field to produce the EA
- Typically the address field is treated as a twos complement number for this operation
- Thus the effective address is a displacement relative to the address of the instruction

Exploits the concept of locality

Saves address bits in the instruction if most memory references are relatively near to the instruction being executed

Basic Addressing Modes: Base Register Addressing

- The referenced register contains a main memory address and the address field contains a displacement from that address
- The register reference may be explicit or implicit
- Exploits the locality of memory references
- Convenient means of implementing segmentation
- In some implementations a single segment base register is employed and is used implicitly
- In others the programmer may choose a register to hold the base address of a segment and the instruction must reference it explicitly

Basic Addressing Modes: Indexing

- The address field references a main memory address and the referenced register contains a positive displacement from that address
- The method of calculating the EA is the same as for base-register addressing
- An important use is to provide an efficient mechanism for performing iterative operations
- Auto-indexing
  - Automatically increment or decrement the index register after each reference to it

  EA = A + (R)

  (R) = (R) +1 or (R) = (R) - 1
- Post-indexing

  EA = A + (R)

  (R) = (R) +1
- Pre-indexing

  (R) = (R) + 1

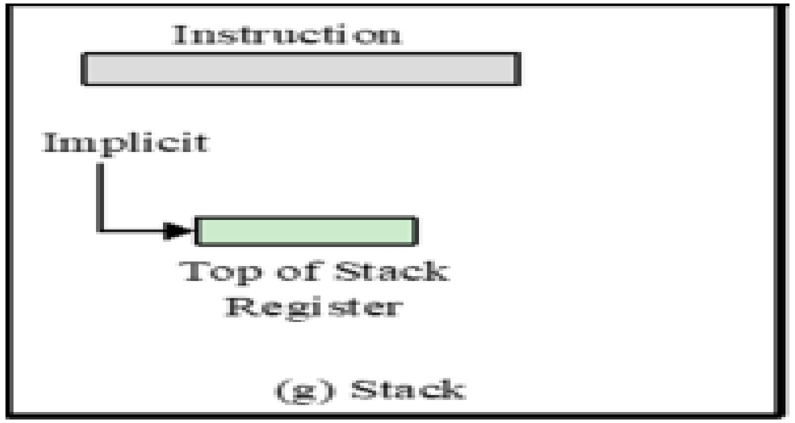
  EA = A + (R)

Basic Addressing Modes: Stack

- A stack is a linear array of locations
  - Sometimes referred to as a *pushdown list* or *last-in-first-out queue*
- A stack is a reserved block of locations
  - Items are appended to the top of the stack so that the block is partially filled
- Associated with the stack is a pointer whose value is the address of the top of the stack
  - The stack pointer is maintained in a register
  - Thus references to stack locations in memory are in fact register indirect addresses

- Is a form of implied addressing

- The machine instructions need not include a memory reference but implicitly operate on the top of the stack



Basic Addressing Modes

Mode	Algorithm	Principal Advantage	Principal Disadvantage
Immediate	Operand = A	No memory reference	Limited operand magnitude
Direct	EA = A	Simple	Limited address space
Indirect	EA = (A)	Large address space	Multiple memory references
Register	EA = R	No memory reference	Limited address space
Register indirect	EA = (R)	Large address space	Extra memory reference
Displacement	EA = A + (R)	Flexibility	Complexity
Stack	EA = top of stack	No memory reference	Limited applicability