Addressing Mode Summary

Mode	Definition/Operation	Processor Requirements	Addressing fixed at
Implied	No addressing needed. The instruction implies where the data comes from. Processors with a single accumulator typically "imply" the accumulator	Varies	N/A
Constant	The data is included as part of the instruction itself. Data is fixed at compile time	A buffer to contain the data	Compile time
Register	The instruction refers to one or more processor registers where it will find the operand	More than one processor register	Compile time
Direct	The operand field of the instruction is an address where data will be found	Memory address register and memory buffer register	Compile time
Register Indirect	The instruction refers to a processor register where it will find the memory address (pointer) where it will in turn find the data	Register capable of containing address and memory buffer register	Register selection fixed at compile time; address at run time
Indirect Addressing	The operand field of the instruction points to an address which contains a second address pointing to the data	Mechanism to handle multiple memory fetches/instruction	Pointer address fixed at compile time; address at run time
Indexed Addressing	Requires two operands: a register containing a base address and either a constant or a second register containing an offset. May be combined with autoindexing. Useful when working with arrays or strings	Mechanism to handle multiple memory fetches/instruction along with addition	Register(s) and constant offset fixed at compile time; address at run time
Base + Indirect Offset	Requires two operands: a register containing a base address and a pointer to address containing an offset. Useful when working with data structures	Mechanism to handle multiple memory fetches/instruction along with addition	Register and pointer to offset fixed at compile time; offset and address at run time
Base + Offset + Indirect Offset	Requires three operands: a register containing a base address, either a constant or a second register containing 1st offset, and a pointer to address containing 2nd offset. Useful when working with arrays of data structures	Mechanism to handle multiple memory fetches/instruction along with addition	Register(s)/constant and pointer to offset fixed at compile time; offset and address at run time
Relative	The operand is a signed offset to be added to the program counter allowing for jumps relative to current instruction	Addition circuitry to conditionally add constant to PC	Signed offset set at compile time; branch decision at run time
Stack	Automatically incremented/decremented pointer to memory for store or retrieve	A dedicated pointer with auto increment/decrement feature	Register being pushed/popped is fixed at compile time

Addresses per Instruction

A typical arithmetic or logical operation has 3 references: 2 sources and a result

3 Address Instructions	2 Address Instructions	1 Address Instructions	0 Address Instructions
• Both operands and the destination for the result are explicitly contained in the instruction word • Example: ADD X, Y, Z performs the operation X = Y + Z • With memory speeds (due to caching) approaching the speed of the processor, this gives a high degree of flexibility to the compiler • Memory = large register set • Very long instruction words	• One addresses used to specify both a source and a destination • Mimics the C programming language syntax A += B • Example: ADD X, Y performs the operation X = X + Y • Very common in instruction sets	• One storage location is addressed while other is implied • Referred to as accumulator-based operation • Example: ADDA X adds X to accumulator A	• Referred to as stack-based operation or "stack machine" • An operation that uses two operands removes top two items from stack, then places result on stack • An operation that uses one operand removes top item from stack, then places result on stack • Interact with the stack using push and pop operations

Example: Y = (A-B) / (C+D*E)
 	
SUB Y,A,B
MUL T,D,E
ADD T,T,C
DIV Y,Y,T
	Example: Y = (A-B) / (C+D*E)
	
	MOV Y,A
	SUB Y,B
	MOV T,D
	MUL T,E
	ADD T,C
	DIV Y,T
		Example: Y = (A-B) / (C+D*E)
		
		LOAD D
		MUL E
		ADD C
		STORE Y
		LOAD A
		SUB B
		DIV Y
		STORE Y
			Example: Y = (A-B) / (C+D*E)
			
			PUSH A
			PUSH B
			SUB
			PUSH C
			PUSH D
			PUSH E
			MUL
			ADD
			DIV
			POP Y

Fewer addresses in the instruction results in:

- More primitive instructions
- Less complex CPU
- Instructions with shorter length – fit more into memory
- More total instructions in a program
- Longer, more complex programs
- Faster fetch/execution of instructions

Addressing Effects on Performance

- There is a relation between operand size, addressing complexity, and instruction execution time
- Example: base + offset + indirect offset addressing mode requires
  - four fetches for instruction and operands
  - two additions to calculate data source address
  - final data fetch
  - operation
  - data store
- Complex addressing may stall execution