Computer Organization
(Instruction set Architecture & Assembly Language Programming)

KR Chowdhary
Professor & Head
Email: kr.chowdhary@gmail.com
webpage: krchowdhary.com

Department of Computer Science and Engineering
MBM Engineering College, Jodhpur

November 14, 2013

Instruction set principles

- Instruction set architecture: A portion of the computer visible to machine language programmer / compiler writer.

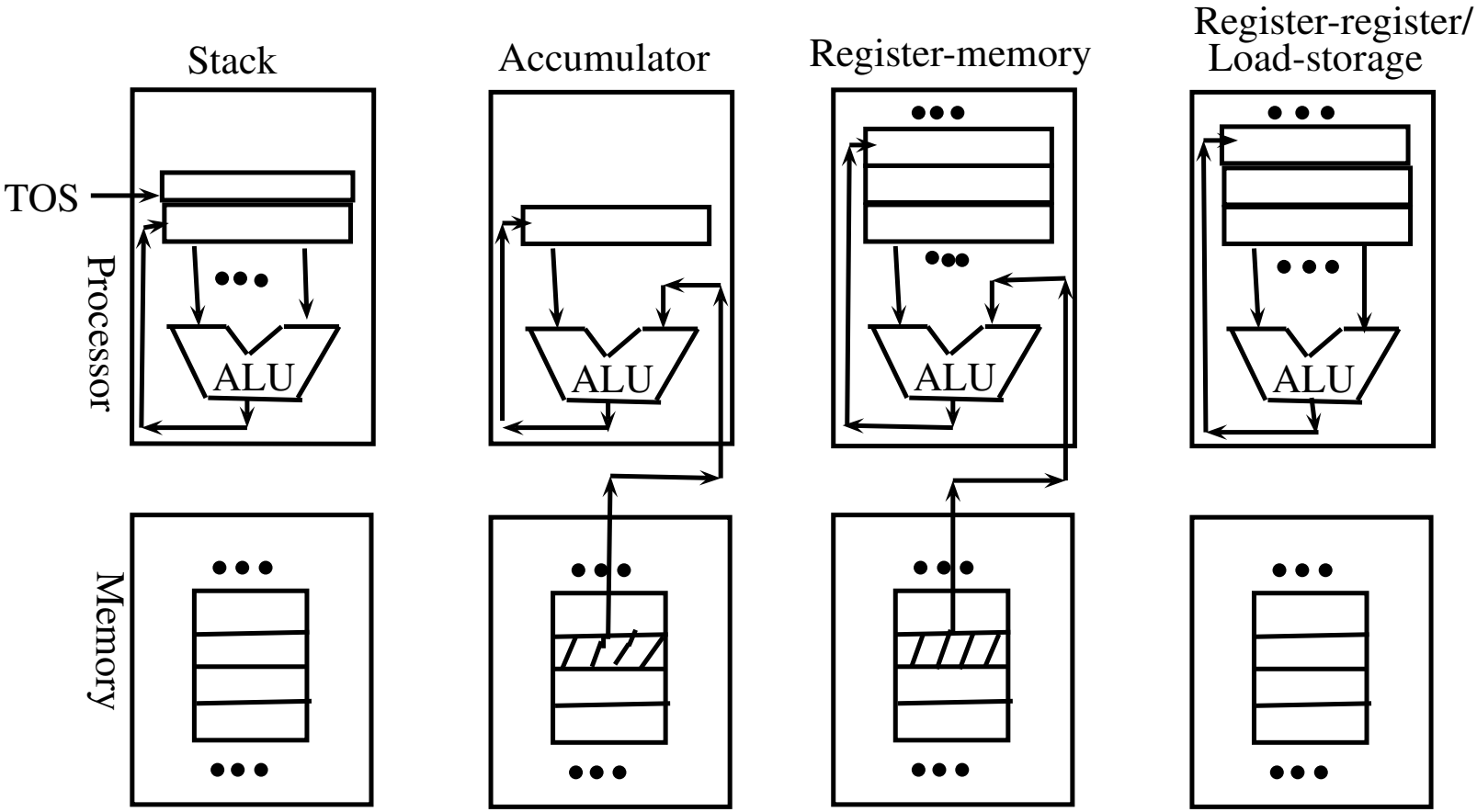
  Execution Environments:

- Desktop: performance of programs decided by integer and floating-point arithmetics, no concern to power consumption, and program size, Appl. Main use web browsing, limited computations. Compiler generated code.
- Servers: data, file servers, web applications, time sharing applications for many users,
- Real-time and embedded systems : Low cost and power, small code size, e.g., DSP (digital signal processing) and media processors, continuous streaming of data, fast execution of code (targeting the worst case performance), code is hand optimized.

Classification of Instruction set Architectures

- High performance Systems: RISC Architecture.
- Architectures:
  1. **Stack architecture**: operand is implicitly on top of stack
  2. **Accumulator architecture**: one operand specified other in accumulator
  3. **General purpose register(GPR) architecture**: operands explicitly in register or memory.
  4. **Memory-Memory Architecture**: All operands are in memory.
- GPR: Registers are faster, and more efficient to use by the compiler.
- In $(A*B)-(B*C)-(A*D)$, multiplication can be evaluated in any order by GPR but not by stack m/c (example later on)

Operand locations for four Instruction set architectures



Instruction set principles

```
\\ compute C = A + B
Stack:       Accumulator:  Register-memory
push A       Load A        //accesses memory as part of
push B       Add B         // instruction
Add          store C       Load R1, A
Pop C                      Add R3, R1, B
                           Store R3, C

Register-register:
//accesses memory through load/store inst.
Load R1, A
Load R2, B
Add  R3, R1, R2
Store R3, C

Memory-memory architecture?
```

Stack v/s Register Memory addressing

Evaluate: (A*B)-(B*C)-(A*D)

Stack Machine:

PUSH A
PUSH B
MULT
PUSH B
PUSH C
MULT
SUB
PUSH A
PUSH D
MULT
SUB
POP T

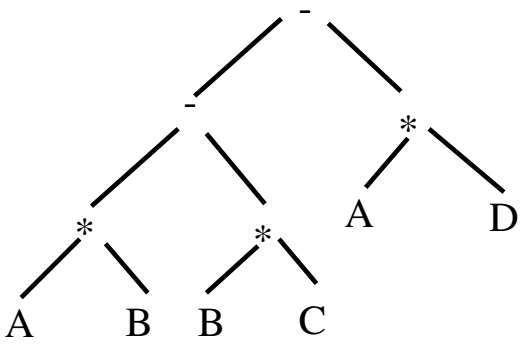


Figure 1: Tree for $(A*B)-(B*C)-(A*D)$. Post-Order: $AB*BC*-AD*-$

Register-Memory Addressing:

MULT E, A, B; A-F registers
MULT F, B, C
SUB F, E, F
MULT E, A, D
SUB T, F, E; T is memory location

Issues in Instruction set Design

- Variables allocated to registers $\Rightarrow$ memory traffic reduces, program speeds up, code density reduces (register named with fewer bits)
- What can be the +ve and -ve issues of R-R, R-M, and M-M instructions?

  Issues in Memory Addressing:
- How memory addresses are specified and interpreted?
- Memory Issue: Big Endian v/s Little Endian. In first, the bits are 0, 1, ..., 6, 7. In second: 7, 6, ..., 1, 0. How does it make difference?

  Instruction formats: Most instructions specify a register transfer operation of the form: an opcode followed with a set of n operands, e.g., $X_1 = f(X_1, X_2, \ldots, X_n)$.
- Addressing Modes: How the architecture specify the address of an object?

Typical Architecture: Intel 8085 Bus structure

- 8-bit CPU
- Communicates with other units through 16-bit address bus, 8-bit data bus, and control bus
- Address: $A_0 - A_{15}$, total addressable memory$=2^{16} = 65536$ (64k). Address locations 0 - 65535 (0000H - FFFFH).
- Databus $D_0 - D_7$ (little E.), multiplexed with lower 8 bits ($A_0 - A_7$) of address bus ($A_0 - A_{15}$).
- Control bus: Various signal lines (binary) carrying signals like Read/write, Enable, Ready, Flag bits, etc.

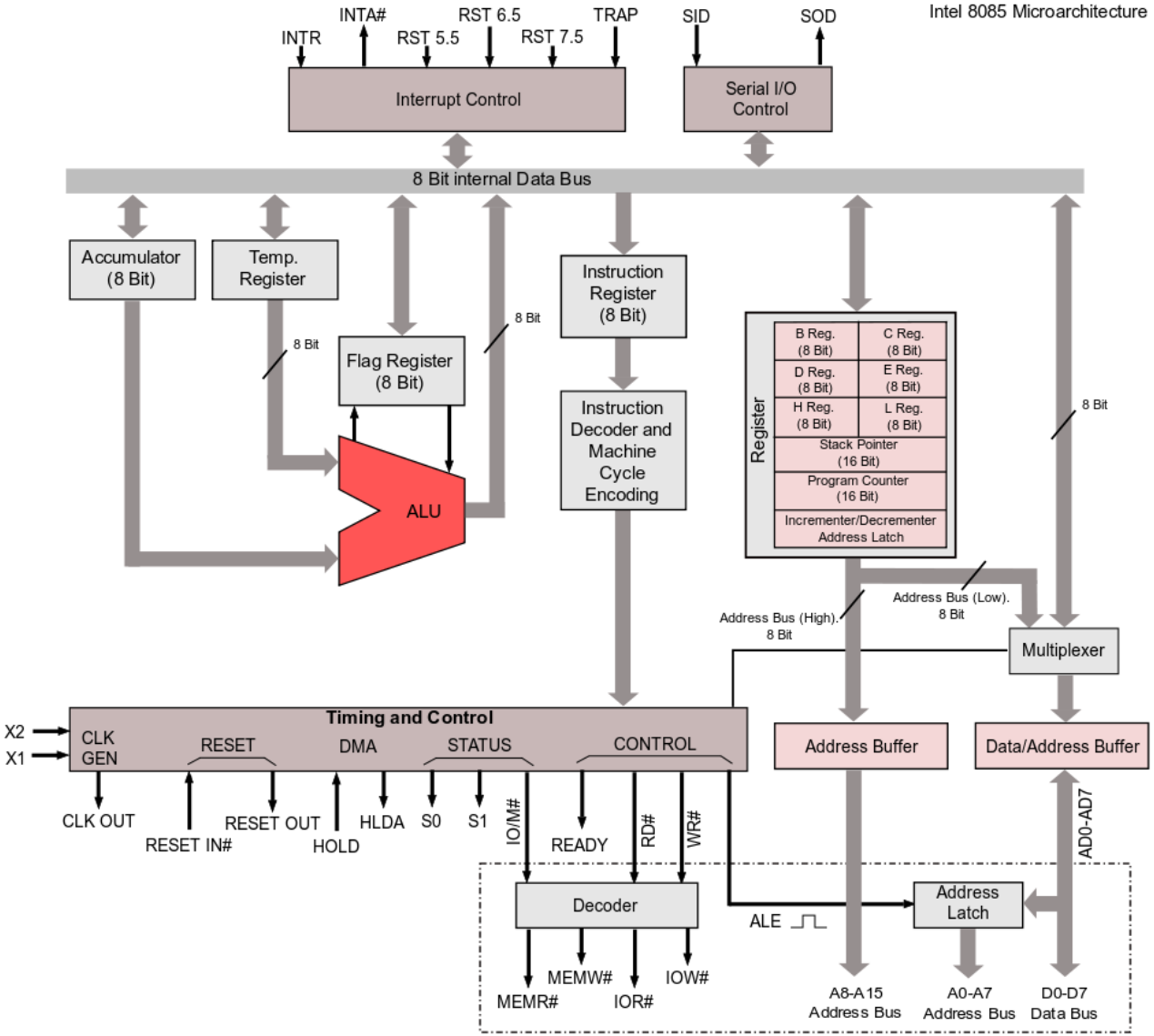
Typical Architecture: Intel 8085 Internal architecture

- 8-bit microprocessor (word length = 8-bit)
- Stores 8-bit data (registers, accumulator, memory locations)
- Performs arithmetic, logic, and data movement operations using 8-bits
- Tests for conditions (if/then)
- Sequence the execution of instructions (jumps, etc)
- Stores temporary data in RAM & register during runtime

Intel 8085 Registers

- ACC + 6 general purpose registers (8-bit): A(111), B(000), C(001), D(010), E(011), H(100), L(101), which can be used to form 3 no. of 16-bit registers, BC(00), DE(01), HL(10), SP(11): two bits in 1st byte.
- Accumulator + Flag register = PSW (processor status register) (status: Z, S, P, C, AC)
- Flag bits: To indicate the result of condition: C(carry), Z(zero), S(sign minus), P(sign plus), AC(auxiliary carry)
- Flag bits are used as Tests for conditions (if/then)
- Program Counter (PC): Contains memory address of next instruction
- Stack Pointer(SP): holds the return address for subroutine call, can save registers(PUSH, POP Instructions)

Intel 8085 Architecture



Intel 8085 assembly language programming

```
;Program to add two numbers:
   MVI A, 7BH
   MVI B, 67H
   ADD B
   HLT

;Program to multiply a given no. by number 4:
   MVI A, 30H
   RRC
   RRC
   MOV B, A
   HLT
```

Intel 8085 assembly language programming

```
;Find greater of two numbers:
     MVI B, 30H
     MVI C, 40H
     MOV A, B
     CMP C
     JZ eq
     JP gt
     JM lt
eq:  MVI D, 00H
     JMP stop
gt:  MVI D, 01H
     JMP stop
lt:  MVI D, 02H
stop: HLT
```

Intel 8085 Instruction set

BRANCH CONTROL GROUP

Jump

JMP adr	C3
JNZ adr	C2
JZ adr	CA
JNC adr	D2
JC adr	DA
JPO adr	E2
JPE adr	EA
JP adr	F2
JM adr	FA
PCHL	E9

Call

CALL adr	CD
CNZ adr	C4
CZ adr	CC
CNC adr	D4
CC adr	DC
CPO adr	E4
CPE adr	EC
CP adr	F4
CM adr	FC

Return

RET	C9
RNZ	C0
RZ	C8
RNC	D0
RC	D8
RPO	E0
RPE	E8
RP	F0
RM	F8

Restart

RST	0	C7
	1	CF
	2	D7
	3	DF
	4	E7
	5	EF
	6	F7
	7	FF

I/O AND MACHINE CONTROL

Stack Ops

PUSH	B	C5
	D	D5
	H	E5
	PSW	F5
POP	B	C1
	D	D1
	H	E1
	PSW*	F1
XTHL		E3
SPHL		F9

Input/Output

OUT byte	D3
IN byte	DB

Control

DI	F3
EI	FB
NOP	00
HLT	76

New Instructions (8085 Only)

RIM	20
SIM	30

ASSEMBLER REFERENCE

Operators

()
NUL
LOW, HIGH
*,/, MOD, SHL, SHR
+, —
NOT
AND
OR, XOR

ASSEMBLER REFERENCE (Cont.)

Pseudo Instruction

General:

ORG
END
EQU
SET
DS
DB
DW

Macros:

MACRO
ENDM
LOCAL
REPT
IRP
IRPC
EXITM

Relocation:

ASEG	NAME
DSEG	STKLN
CSEG	STACK
PUBLIC	MEMORY
EXTRN	

Conditional Assembly:

IF
ELSE
ENDIF

Constant Definition

0BDH, 1AH	Hex
105D, 105	Decimal
72O, 72Q	Octal
11011B, 00110B	Binary
'TEST', 'A' 'B'	ASCII

ARITHMETIC AND LOGICAL GROUP

Add*

ADD	A	87
	B	80
	C	81
	D	82
	E	83
	H	84
	L	85
	M	86
ADC	A	8F
	B	88
	C	89
	D	8A
	E	8B
	H	8C
	L	8D
	M	8E

Subtract*

SUB	A	97
	B	90
	C	91
	D	92
	E	93
	H	94
	L	95
	M	96
SBB	A	9F
	B	98
	C	99
	D	9A
	E	9B
	H	9C
	L	9D
	M	9E

Double Add †

DAD	B	09
	D	19
	H	29
	SP	39

Increment**

INR	A	3C
	B	04
	C	0C
	D	14
	E	1C
	H	24
	L	2C
	M	34
INX	B	03
	D	13
	H	23
	SP	33

Decrement**

DCR	A	3D
	B	05
	C	0D
	D	15
	E	1D
	H	25
	L	2D
	M	35
DCX	B	0B
	D	1B
	H	2B
	SP	3B

Specials

DAA*	27
CMA	2F
STC†	37
CMC†	3F

Rotate †

RLC	07
RRC	0F
RAL	17
RAR	1F

Logical*

ANA	A	A7
	B	A0
	C	A1
	D	A2
	E	A3
	H	A4
	L	A5
	M	A6
XRA	A	AF
	B	A8
	C	A9
	D	AA
	E	AB
	H	AC
	L	AD
	M	AE
ORA	A	B7
	B	B0
	C	B1
	D	B2
	E	B3
	H	B4
	L	B5
	M	B6
CMP	A	BF
	B	B8
	C	B9
	D	BA
	E	BB
	H	BC
	L	BD
	M	BE

Arith & Logical Immediate

ADI byte	C6
ACI byte	CE
SUI byte	D6
SBI byte	DE
ANI byte	E6
XRI byte	EE
ORI byte	F6
CPI byte	FE

DATA TRANSFER GROUP

Move

MOV	A,A	7F
	A,B	78
	A,C	79
	A,D	7A
	A,E	7B
	A,H	7C
	A,L	7D
	A,M	7E
MOV	B,A	47
	B,B	40
	B,C	41
	B,D	42
	B,E	43
	B,H	44
	B,L	45
	B,M	46
MOV	C,A	4F
	C,B	48
	C,C	49
	C,D	4A
	C,E	4B
	C,H	4C
	C,L	4D
	C,M	4E
MOV	D,A	57
	D,B	50
	D,C	51
	D,D	52
	D,E	53
	D,H	54
	D,L	55
	D,M	56

Move (cont)

MOV	E,A	5F
	E,B	58
	E,C	59
	E,D	5A
	E,E	5B
	E,H	5C
	E,L	5D
	E,M	5E
MOV	H,A	67
	H,B	60
	H,C	61
	H,D	62
	H,E	63
	H,H	64
	H,L	65
	H,M	66
MOV	L,A	6F
	L,B	68
	L,C	69
	L,D	6A
	L,E	6B
	L,H	6C
	L,L	6D
	L,M	6E
MOV	M,A	77
	M,B	70
	M,C	71
	M,D	72
	M,E	73
	M,H	74
	M,L	75
XCHG		EB

Move Immediate

MVI	A, byte	3E
	B, byte	06
	C, byte	0E
	D, byte	16
	E, byte	1E
	H, byte	26
	L, byte	2E
	M, byte	36

Load Immediate

LXI	B, dble	01
	D, dble	11
	H, dble	21
	SP, dble	31

Load/Store

LDAX B	0A
LDAX D	1A
LHLD adr	2A
LDA adr	3A
STAX B	02
STAX D	12
SHLD adr	22
STA adr	32

byte = constant, or logical/arithmetic expression that evaluates to an 8-bit data quantity. (Second byte of 2-byte instructions).
dble = constant, or logical/arithmetic expression that evaluates to a 16-bit data quantity. (Second and Third bytes of 3-byte instructions).
adr = 16-bit address (Second and Third bytes of 3-byte instructions).
* = all flags (C, Z, S, P, AC) affected.
** = all flags except CARRY affected; (exception: INX and DCX affect no flags).
† = only CARRY affected.

Intel 8085 addressing modes

- Immediate addressing (MVI B, 25H)
- Direct addressing (LDA 1020H)
- Register addressing (MOV B, C)
- Implied addressing (CMA, RAR)
- Register Indirect addressing(MOV A, M; ADD M).
- Register Indirect addressing(LDAX B, LDAX D, STAX B, STAX D)

Programming in assembly language

```
;Code to sum five locations and store the result at subsequent
;location:
      LXI H, 1010H ; memory pointer for start of data
      MVI C, 05H ; initialize counter value
      XRA A ; Exclusive OR A with itself
loop: ADC M ; add memory into Accumulator with carry
      INX H ; increment the memory pointer register pair HL
      DCR C ; decrement the counter
      JNZ loop ; if counter not zero then repeat the loop
      MOV M, A ; store the sum at subsequent location.
      HLT ; otherwise halt the processor
```

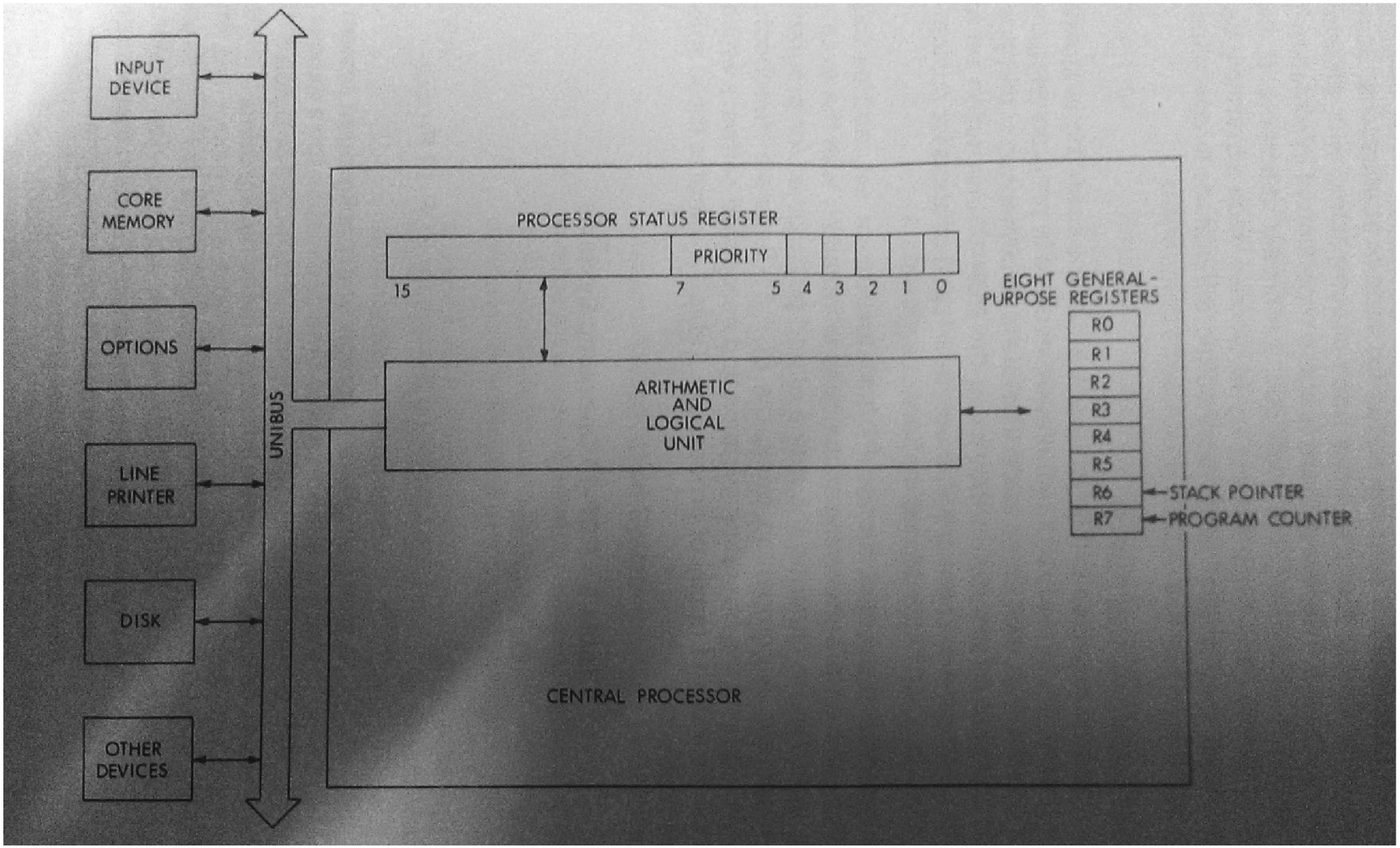
Programming in assembly language

```
;Code to copy paste 1000H to 2000H for 25H locations:
       LXI B, 1010H ; source pointer
       LXI D, 2000H ; destination pointer
       MVI H, 25H ; counter initialize
 loop: LDAX B
       STAX D
       INX B
       INX D
       DCR H
       JNZ loop
       HLT
```

PDP-11 Mini-Computer, 1970s

- Registers: R0:R5, SP=R6, PC=R7,(all 16 bits), Status Flags: I, V, N, Z, C
- Address: 16-bits (64K), (32-k words)
- additional instructions, like MUL, DIV, WAIT, RESET, and many more powerful instructions
- later versions supported Virtual memory.
- 8085 instruction set is subset of PDP11 (DEC machine)
- Addressing modes: register, autoincrement, autodecrement, index, indirect, immediate, absolute, relative

PDP-11


INPUT DEVICE
CORE MEMORY
OPTIONS
LINE PRINTER
DISK
OTHER DEVICES
UNIBUS
PROCESSOR STATUS REGISTER
PRIORITY
15
7
5 4 3 2 1 0
EIGHT GENERAL-PURPOSE REGISTERS
R0
R1
R2
R3
R4
R5
R6
R7
STACK POINTER
PROGRAM COUNTER
ARITHMETIC AND LOGICAL UNIT
CENTRAL PROCESSOR

IBM360 Addressing Modes

16 General Purpose Register: $R_0 - R_{15}$, Mem adr: 20bits

Register	Add R3, R4
Immediate	Add R4, #3
Displacement	Add R4, 100(R1)
Register Indirect	Add R4, (R1)
Indexed	Add R3, (R1+R2); R1 base, R2 Index
Direct/Absolute	Add R1, (1001); [R1] ← [R1]+m[1001]
Memory Indirect	Add R1, @(R3); [R1]←[R1]+M[M[R3]]
Auto Increment	Add R1, (R2)+; [R1]←[R1]+M[R2]
	[R2]← [R2]+d; d size of elem.
Auto decrement	Add R1, -(R2);[R2]← [R2]-d; R[1]←[R1]+M[R2]

- Addressing modes reduce the instruction count, add complexity in building computer, may increase average clock cycles per instruction.

  Number of addresses:
- CDC 6600: ADD Z, Y, X ; three address

1. Fever operands → lesser functions per instruction → longer programs → longer execution times.
2. long instructions with multiple operands → more complex decoding & processing circuits.

Exercises

1. Assume an instruction set that uses a fixed 16-bit instruction length. Operand specifiers are 6 bits in length. There are 5 two operand instructions and 33 zero operand instructions. What is the maximum number of one-operand instructions that can be encoded using the fixed 16-bit instruction length?
2. A given processor has 32 registers, uses 16-bit immediate and has 142 instructions. In a given program,
   - 20 % of the instructions take 1 input register and have 1 output register.,
   - 30 % have 2 input registers and 1 output register,
   - 25 % have 1 input register, 1 output register and take an immediate input as well, and the remaining 25 % have one immediate input and 1 output register.

Exercises

2.1 For each of the 4 types of instructions , how many bits are required? Assume that it requires that all instructions be a multiple of 8 bits in length.

2.2 How much less memory does the program take up if variable-length instruction set encoding is used as opposed to fixed-length encoding?

3. Compare the memory efficiency of the following instruction set architectures:
   - Accumulator- All operations occur between a single register and a memory location. There are two accumulators of which one is selected by the opcode;
   - Memory-memory: All instruction addresses reference only memory locations
   - Stack - All operations occur on top of the stack. The implementation uses a hardwired stack for only the top two stack entries, which keeps the processor circuit very small and low cost. Additional stack positions are kept in memory locations, and accesses to these stack positions require memory references.
   - Load-store - All operations occur in registers, and register-to register instructions have three register names per instruction.

Exercises

To measure memory efficiency, following are assumptions about all 4 instruction sets:

- All instructions are an integral number of 8-bit in length;
- The opcode is always 8 bits;
- Memory accesses use direct address
- The variables A, B, C, and D are initially in memory

a. Invent your own assembly language mnemonics and for each architecture write the best equivalent assembly language code for this high level language code:
   A = B + C;
   B = A + C;
   D = A - B;

b. Assume the given code sequence is from a small, embedded computer application, such as a microwave oven controller that uses 16-bit memory addresses and data operands. If a load-store architecture is used, assume that it has 16 general-purpose registers. Answer the following questions:

Exercises

- How many instruction bytes are fetched?
- How many bytes of data are transferred from/to memory?
- Which architecture is the most efficient as measures in code size?
- Which architecture is most efficient as measured by total memory traffic (code + data)

4. Specify the register contents and the flag status as the following instructions are executed:

```
XRA A
MVI B, FFH
INR B
DCR A
ADD B
SUI 86H
ANA C
RST1
```

Exercises

5. A system is designed to monitor the temperature of a furnace. Temperature readings are recorded in 16 bits and stored in memory locations starting at 7060H. The high-order byte is stored first and the low-order byte is stored in the next consecutive memory location.However, the low-order byte of all the temperature readings is constant. Write 8085 ALP to transfer the high-order readings to consecutive memory locations starting at 7080H and discard the low-order bytes.Temperature Readings (H): 6745, 8745, 1F45, 3045, 8045, 7F45.
6. First set of data is stored from memory locations starting from 6155H to 6165H. Second set of data is stored from 6255H to 6265H. Write 8085 ALP to interchange the contents of memory locations with each other.
   (6155H) <——————————> (6255H)
   (6165H) <——————————-> (6265H)
7. Download intel 8085 simulator (gnusim8085), using command:
   $ sudo apt-get install gnusim8085 <enter>,
   and sun the various simulation programs.

Exercises

8. List the assembly language program generated by a compiler from the following Fortran program. Assume integer values of one byte size.

```
SUM = 0
SUM = SUM + A + B
DIF = DIF - C
SUM = SUM + DIF
```

9. List the assembly language program generated by the compiler for the following Fortran IF statement:

```
IF(A - B) 10, 20, 30
```

The program branches to statement 10 if A-B <0; to statement 20 if A-B $=0$; and to statement 30 if A -B > 0.

10. Write a program to multiply two unsigned numbers.

Bibliography

- John L. Hennessy & David A. Patterson, Computer Architecture - A qualitative approach, 3rd Edition, Elsevier (indian print).
- PDP11 processor handbook, Digital Equipment Corporation, 1979.
- http://bitsavers.trailing-edge.com/pdf/intel/MCS80/9800301D_8080_8085_Assembly_Language_Programming_ Manual_May81.pdf
- http://www.iitg.ernet.in/asahu/cs421/Lects/Lec04.pdf