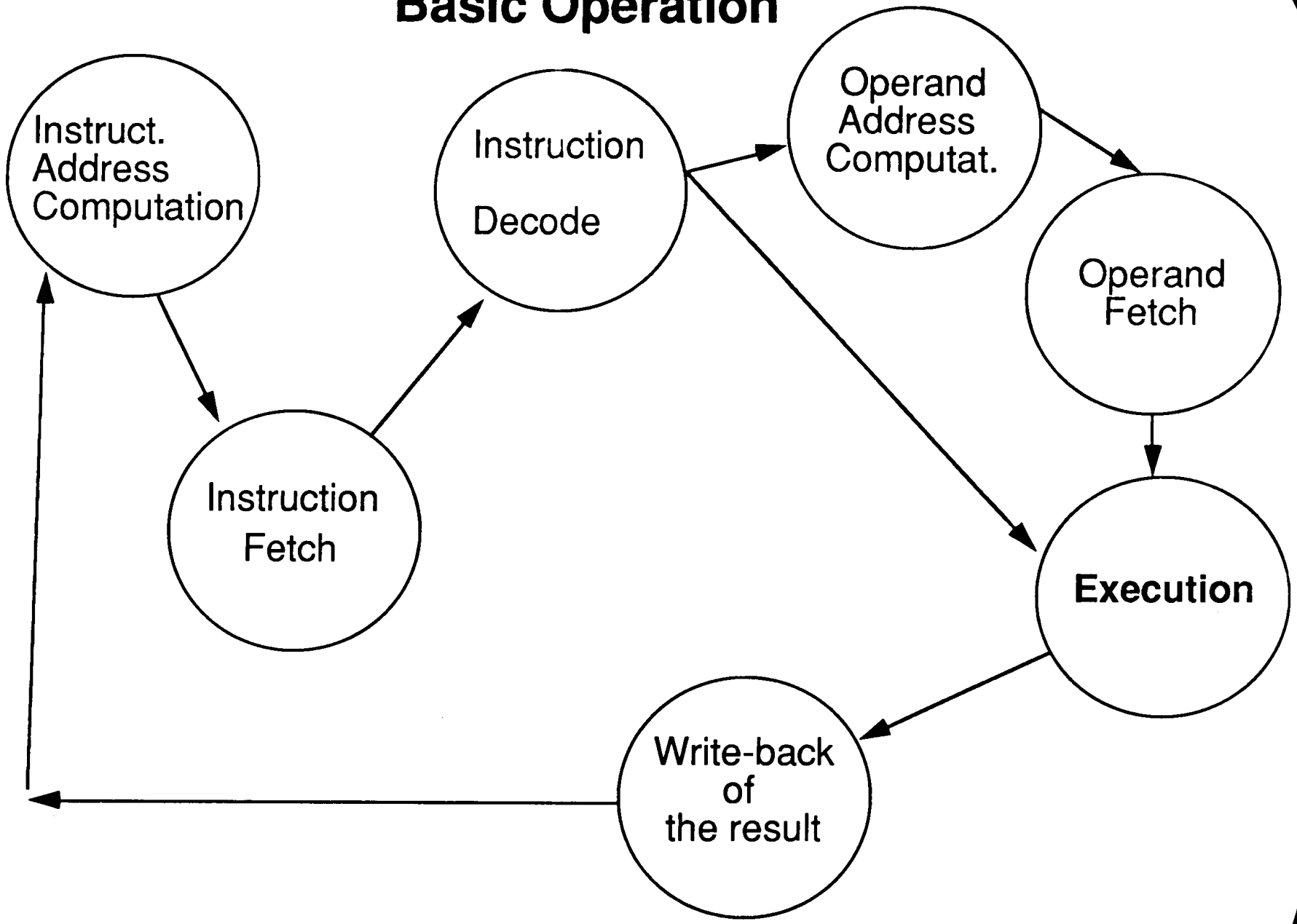
Basic Operation



CPU: Concepts, Organization

Instruction Formats (example):

Load/ Store:

Operation

Branch

CPU: Concepts, Organization

Register - Transfer Operations : Addition Operation ADD

IF
1. MABus <- IAR , R=1
2. MDBus <- Data (instruction)
3. IR <- Data

ID 4. Decode Instruction (set the control lines to ADD)

OF 5. SRA <- RS1, SRB <- RS2

EX 6. SRC <- Result (SRA + SRB)

WB 7. RT <- SRC

8. IAR <- IAR + 4
9. Check for Interrupts

RT ← RS1 + RS2

LD : Load Instruction:

LD	R3	Address

R3 ← M[Address]

Major Cycles:

1. IF : Instruction Fetch
2. ID : Insruction Decode
3. OAG : Operand Address Generation
4. OF : Operand Fetch
5. EXEC : Execution (arithmetic / logical)
6. WB : Write Back (writing of the result)

IF: Instruction Fetch Cycle is the same for all instructions.

We need to check for the Interrupt and detrmine how are we going to handle Interrupt.

CPU: Concepts, Organization

Register - Transfer Operations : Load Operation LD

IF
1. MABus <- IAR R = 1
2. MDBus <- Data (instruction)
3. IR <- Data

ID
4. Decode Instruction

OF
5. SRA <- RB, SRB <- Displacement

DAC
6. SRC <- Address

MA
7. MAR <- SRC
8. MABuss <- MAR
9. MDBus <- Data
10. MDR <- MDBus

WB
11. RT <- MDR

IAC
12. IAR <- IAR + 4
13.. Check for INT

BR (C) Branch Instruction:

BR	CC	Address

Major Cycles:

1. IF : Instruction Fetch
2. ID : Insruction Decode
3. AG : Address Generation
4. Testing of the Condition CC
5. Writing into the IAR

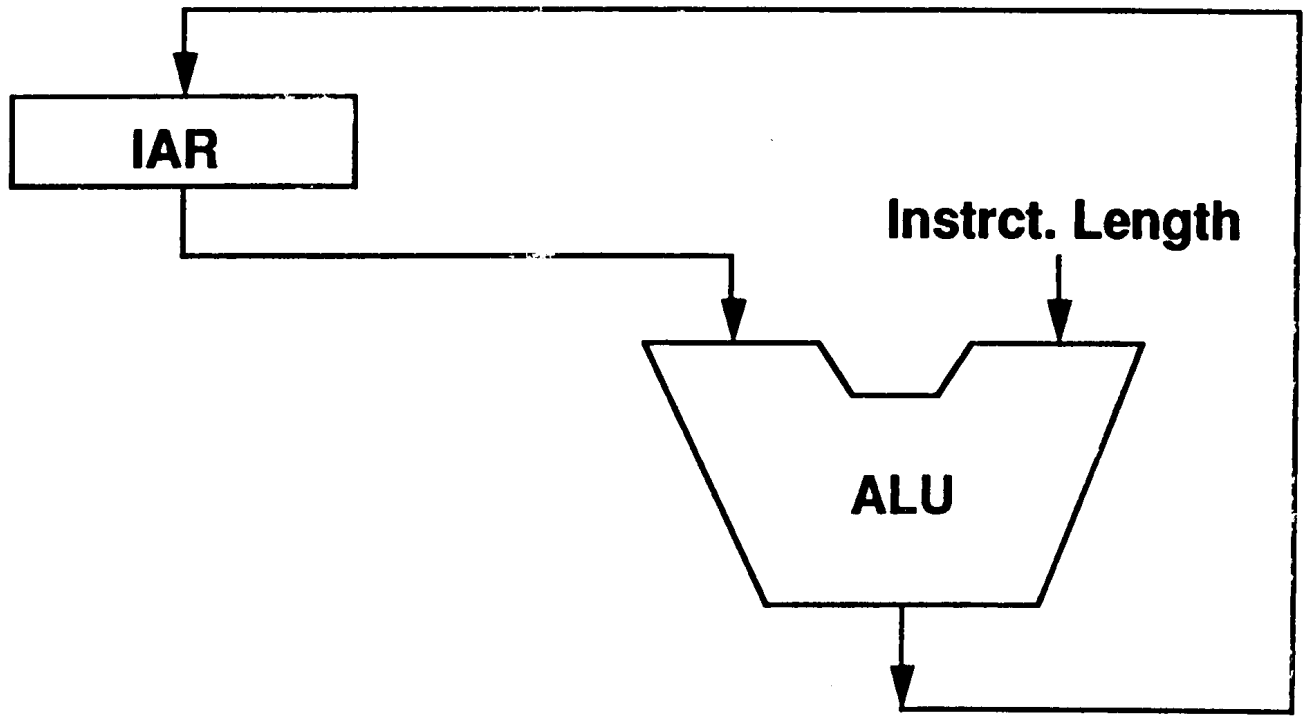
IF : Instructin Fetch Cycle:

1. MA_Bus <-- IAR

2. R/W' <-- 1, Enable IR to read data from the Memory Bus

3. Increment the IAR for the length of the currently decoded instruction:

   IAR <-- IAR + IL (IAR is now pointing to the next instruction)



4. Decide about Interrupt

Register - Transfer Operations : Branch Operation BRC

1. MABus <- IAR
2. MDBus <- Data (instruction)
3. IR <- Data
4. Decode Instruction
5. Examne CC: if CC=1

then: else:

6. IAR <- Address 6a. IAR <- IAR + 4

7. Check for INT