Processor Basics

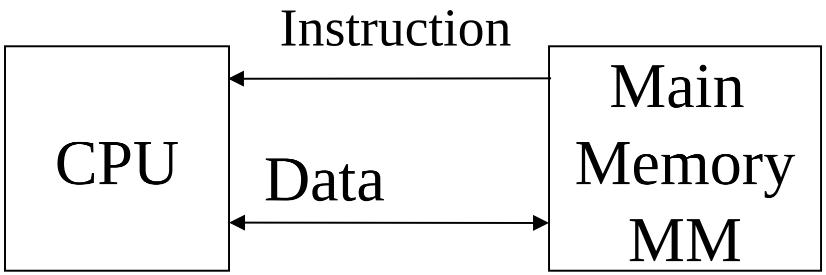
Chapter 3
Book of John P. Hayes

CPU Organization

The primary function of the CPU is to execute sequences of instructions, which are stored in an external main memory. The execution is carried out in three steps:

- ✓ The CPU transfers instruction and when necessary, their input data (operand) from main memory to registers in the CPU.
- ✓ The CPU executes the instruction is their stored sequence.
- ✓ When necessary, the CPU transfers output results from the CPU registers to main memory.

External Communication

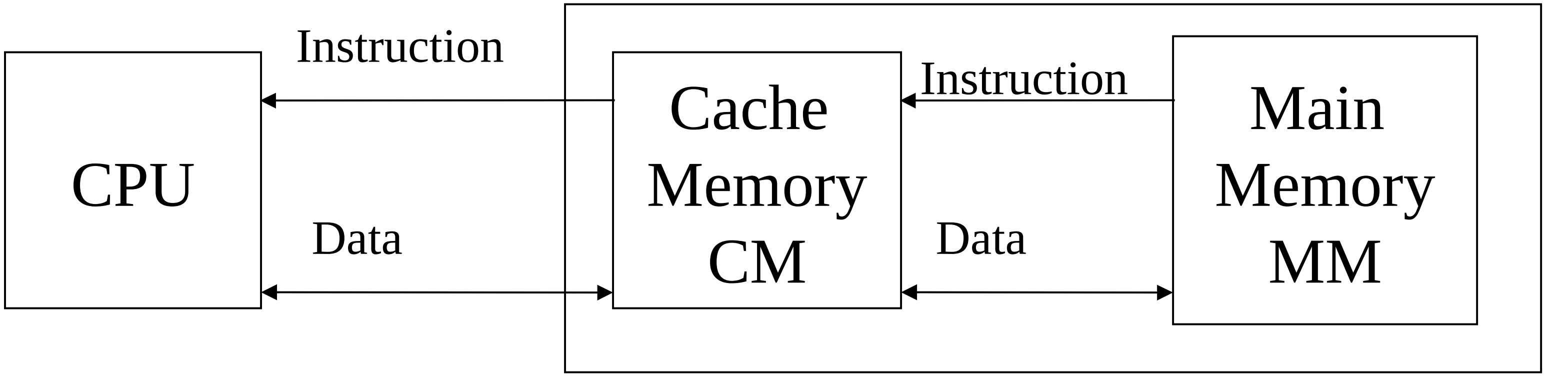

Instruction

CPU

Data

Main Memory MM

(a) Processor-memory communication without a cache


Instruction

CPU

Data

Cache Memory CM

Instruction

Data

Main Memory MM

(b) Processor-memory communication with a cache

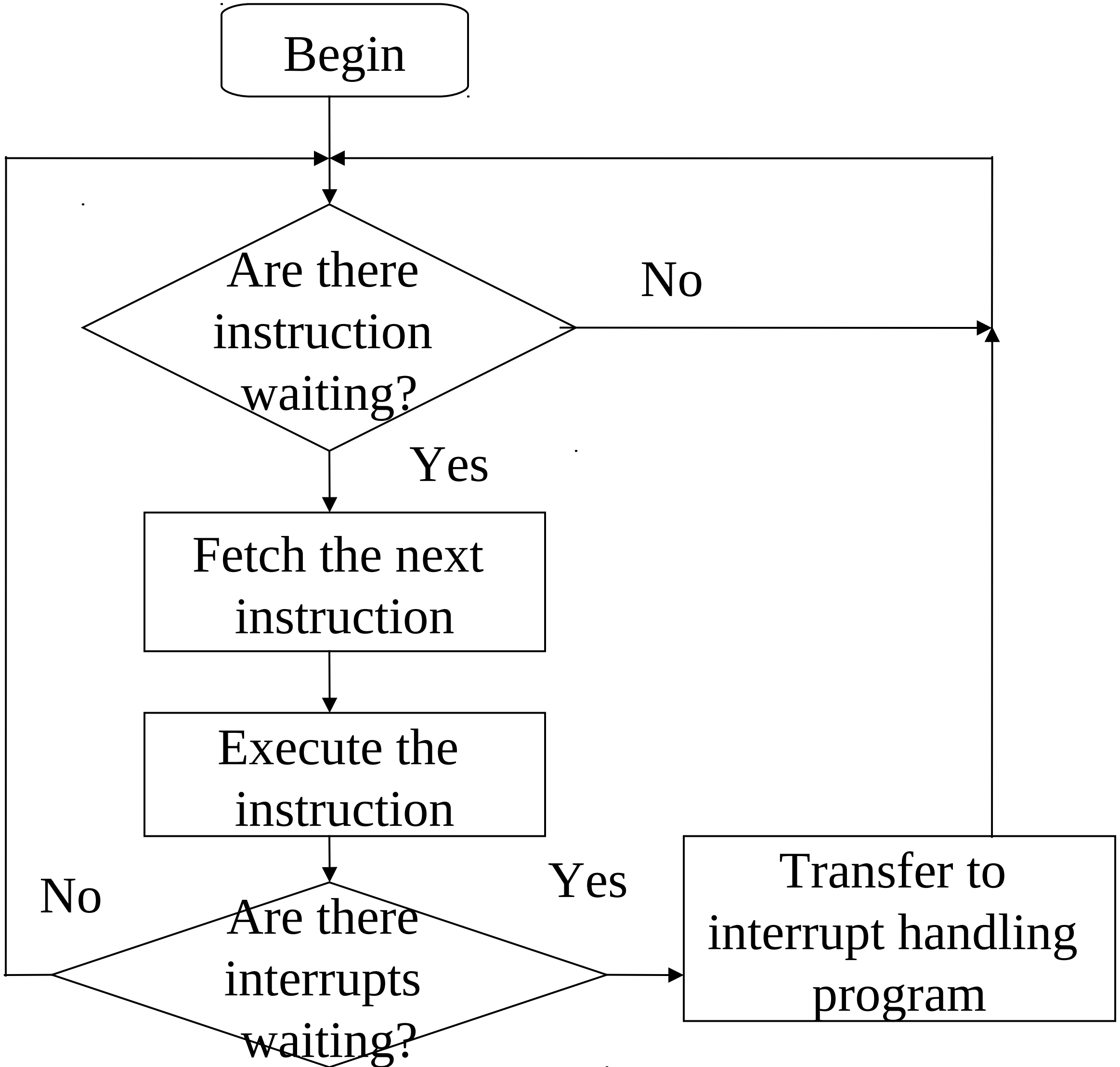
External Communication

- ✓ CPU can perform a memory load or store operation from cache in a single clock cycle. On the other hand, the same operations take many clock cycles if they are performed from main memory.
- ✓ CPU considers the cache and the main memory as a single, seamless memory space consisting of 2^m addressable storage locations.

User and Supervisor Programs

- ✓ A user or application program handles a specific applications.
- ✓ A Supervisor program manages various routine aspects of the computer system. It is typically part of the computer's operating system.
- ✓ For normal operation, CPU continually switches back and forth between user and supervisor programs.
- ✓ The requests for supervisor services from the secondary memory and IO devices are known as interrupt.
- ✓ In the event of interrupt, the CPU suspends execution of the current program and transfer to an appropriate interrupt handling program.
- ✓ CPU frequently check for the presence of interrupt request.

CPU Operation


Begin
Are there instruction waiting?
No
Yes
Fetch the next instruction
Execute the instruction
No
Yes
Are there interrupts waiting?
Transfer to interrupt handling program

CPU Operation

- ✓ The sequence of operations performed by the CPU in processing an instruction constitutes an **instruction cycle**.
- ✓ Processing of all instruction generally requires two steps:

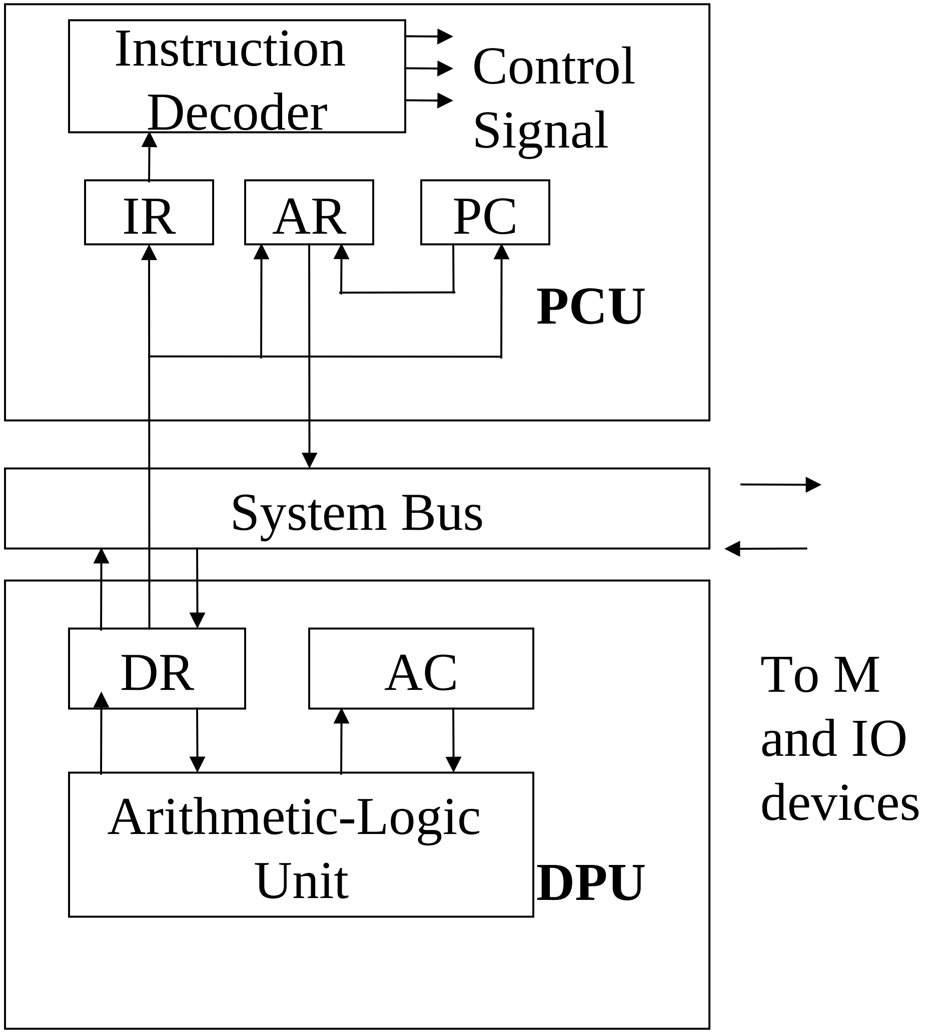
  1. **Fetch step-** a new instruction is read from the external memory.

  2. **Execute step-**The operations specified by the instruction are executed.

CPU Operation

- ✓ The actions of the CPU during an instruction cycle are defined by a sequence of micro operations, each of which involves a register transfer operation.
- ✓ The time required for the shortest well-defined CPU micro operation is the CPU cycle time or clock period T_{clock}.
- ✓ The number of CPU cycles required to process an instruction varies with the instruction type and the extent to which the processing of individual instructions can be overlapped.

Accumulator-based CPU



PCU=Program Control Unit

AR= Address Register

IR=Instruction Register

PC=Program Counter

DPU= Date Processing Unit

AC=Accumulator Register

DR=Data Register

Accumulator-based CPU

- ✓ An instruction that refers to a data word in M contains two parts, an opcode ***op*** and a memory address ***adr*** and ***I=op.adr*.**
- ✓ Each instruction cycle begins with the instruction fetch operation IR.AR = M (PC), where IR=***op*** and AR=***adr***. Here PC contains the address of the current instruction in the memory.
- ✓ Instruction that do not reference M do not use AR. The opcode part specifies the CPU registers to use, as well as the operation to be carried out.
- ✓ Once it has placed the opcode of I in IR, the CPU proceeds to decode and execute it. At this point, the CPU can increment PC in order to obtain the address of the next instruction.

Accumulator-based CPU

- ✓ The load instruction transfers a word from the memory location with address ***adr*** to the accumulator.

  AC := M (adr)

- ✓ The store instruction transfers a word from AC to M.

M (adr) : = AC

Programming Consideration

- ✓ Data processing operations normally requires up to 3 operands. For example, Z:= X + Y.
- ✓ The accumulator based CPU supports only single-address instructions.
- ✓ A program that implement Z:=X+Y, where X, Y and Z all refer to data words in M, can take the following form:

HDL format	Assembly language format	Comment
1. AC:=M(X)	LD X	Load X from M into AC
2. DR:=AC	MOV DR, AC	Move contents of AC to DR
3. AC:=M(Y)	LD Y	Load Y into AC
4. AC:=AC+DR	ADD	Add DR to AC
5. M(Z):=AC	ST Z	Store contents of AC in M

Programming Consideration

- Consider an instruction of form: AC:= f_i (AC, M (adr)) which require to move M(adr) to or from DR and one to perform the operation f_i.
- Memory reference complicates the instruction-decoding logic.
- Overall execution time should be reduced.

HDL format	Assembly language format	Comment
1. AC:=M(X)	LD X	Load X from M to AC
2. AC:=AC+M(Y)	ADD Y	Load Y into DR and add to AC
3. M(Z):=AC	ST Z	Store contents of AC in M

Instruction Set

The instruction set of the accumulator based CPU:

Type	**Instruction**	**HDL format**	**Assembly format**
Data	Load	AC:=M(X)	LD X
Transfer	Store	M(X):= AC	ST X
	Move Register	DR:=AC	MOV DR, AC
Data	Add	AC:=AC+DR	ADD
Processing	Subtract	AC:=AC-DR	SUB
	And	AC:=AC and DR	AND
	Not	AC:= not AC	NOT
Program	Branch	PC:=M (adr)	BRA adr
Control	Branch zero	if AC = 0 then	BZ adr
		PC:=M (adr)	

Instruction Set

- ✓ We use load and store instruction to access memory. This feature is known as load/store architecture.
- ✓ The instruction set is complete.

A Multiplication Program

- ✓ The program should implement AC:=AC×N where, the multiplicand is the initial content of AC and the multiplier N is a variable stored in memory.
- ✓ AC×N is implemented by executing the ADD instructions N times in the form AC+AC+………..+AC
- ✓ The memory location storing N acts as a count register and after each add operation, it is decremented by 1 until it reaches 0.
- ✓ The test for N=0 is performed by means of BZ instruction.
- ✓ The memory locations:

  one – stores constant 1

  mult– store N

  ac – store Y

  prod– partial product

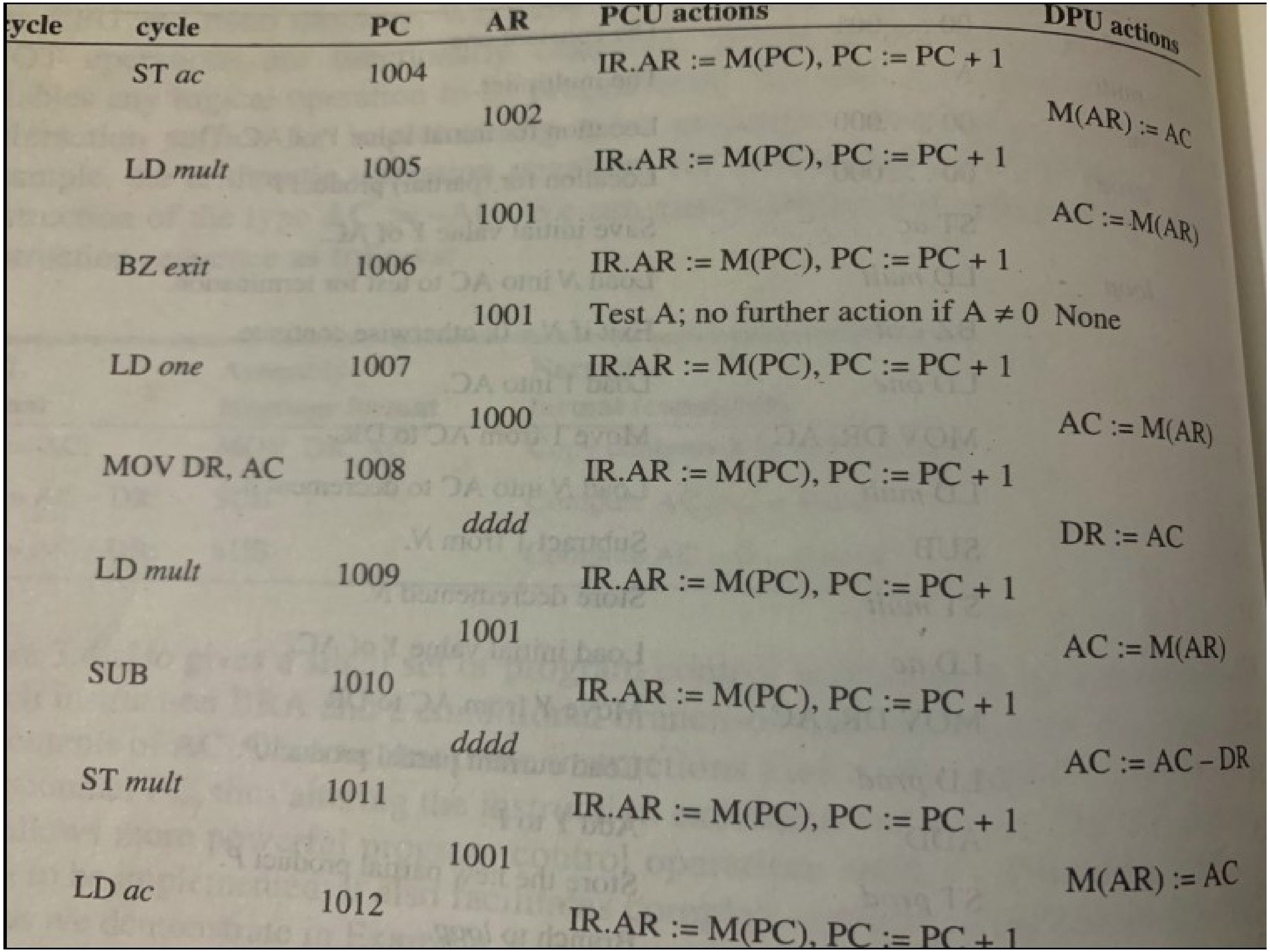
A Multiplication Program

Line	Location	Instruction/ Data	Assembly formatz
0	one	00…………..01	
1	mult	N	
2	ac	00…………00	
3	prod	00…………00	
4		ST ac	
5	Loop	LD mult	AC:= M (mult)
6		BZ exit	If AC= 0 then exit
7		LD one	AC:= M (one)
8		MOV DR, AC	DR:=AC
9		LD mult	AC:= M (mult)
10		SUB	AC:=AC-DR
11		ST mult	M (mult):= AC
12		LD ac	AC:= M (ac)
13		MOV DR, AC	DR:=AC
14		LD prod	AC:= M (prod)
15		ADD	AC:=AC+DR
•		ST prod	M (prod): =AC
1		BRA Loop	
2	exit		

Several Limitations of Accumulator-based CPU

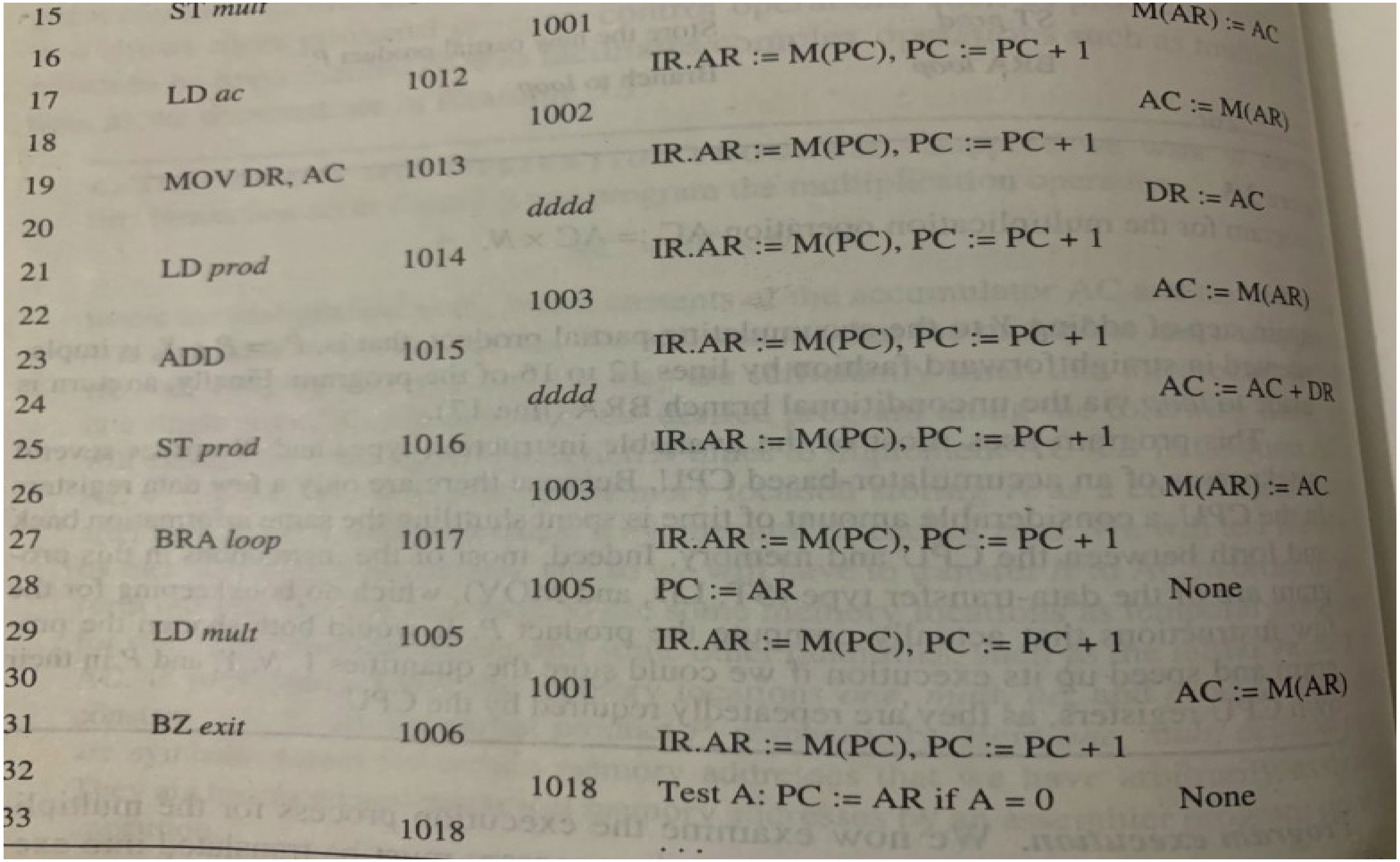
- Because of few data registers in CPU, a considerable amount of time is spent shuttling the same information back and forth between the CPU and memory.
- It would both shorten the program and speed up its execution if it is possible to store the quantities 1, N, Y and P in their own CPU registers, as they are repeatedly required by the CPU.

Program Execution in Accumulator-based CPU



ycle	cycle	PC	AR	PCU actions	DPU actions
	ST *ac*	1004		IR.AR := M(PC), PC := PC + 1	
			1002		M(AR) := AC
	LD *mult*	1005		IR.AR := M(PC), PC := PC + 1	
			1001		AC := M(AR)
	BZ *exit*	1006		IR.AR := M(PC), PC := PC + 1	
			1001	Test A; no further action if A ≠ 0	None
	LD *one*	1007		IR.AR := M(PC), PC := PC + 1	
			1000		AC := M(AR)
	MOV DR, AC	1008		IR.AR := M(PC), PC := PC + 1	
			dddd		DR := AC
	LD *mult*	1009		IR.AR := M(PC), PC := PC + 1	
			1001		AC := M(AR)
	SUB	1010		IR.AR := M(PC), PC := PC + 1	
			dddd		AC := AC – DR
	ST *mult*	1011		IR.AR := M(PC), PC := PC + 1	
			1001		M(AR) := AC
	LD *ac*	1012		IR.AR := M(PC), PC := PC + 1	

Program Execution in Accumulator-based CPU



15	ST *mult*		1001		M(AR) := AC
16					
17	LD *ac*	1012		IR.AR := M(PC), PC := PC + 1	
18			1002		AC := M(AR)
19	MOV DR, AC	1013		IR.AR := M(PC), PC := PC + 1	
20			*dddd*		DR := AC
21	LD *prod*	1014		IR.AR := M(PC), PC := PC + 1	
22			1003		AC := M(AR)
23	ADD	1015		IR.AR := M(PC), PC := PC + 1	
24			*dddd*		AC := AC + DR
25	ST *prod*	1016		IR.AR := M(PC), PC := PC + 1	
26			1003		M(AR) := AC
27	BRA *loop*	1017		IR.AR := M(PC), PC := PC + 1	
28			1005	PC := AR	None
29	LD *mult*	1005		IR.AR := M(PC), PC := PC + 1	
30			1001		AC := M(AR)
31	BZ *exit*	1006		IR.AR := M(PC), PC := PC + 1	
32			1018	Test A: PC := AR if A = 0	None
33		1018		. . .	

Architecture Extension

- Multipurpose register set for storing data and addresses:
  - Replace AC, DR and AR
- Additional data, instruction and address types:
  - Instructions to handle data and addresses with several different word size and formats.
  - Circuits for multiplication and division.
  - Call and return instructions.
- Status or condition code or flag registers:
  - Indicates exceptional condition.
  - Conditional branch instruction can test the status register.

Architecture Extension

- Program Control Stack:
  - Supports transfer of control among programs due to procedure call and external interrupts.
  - Part of external memory is used as stack.
  - Stack pointer keeps track of the content of the stack.

speci...
Special circuits ... also be used for this pur... from the instruction register, whic...

ried
cut
thr
ex

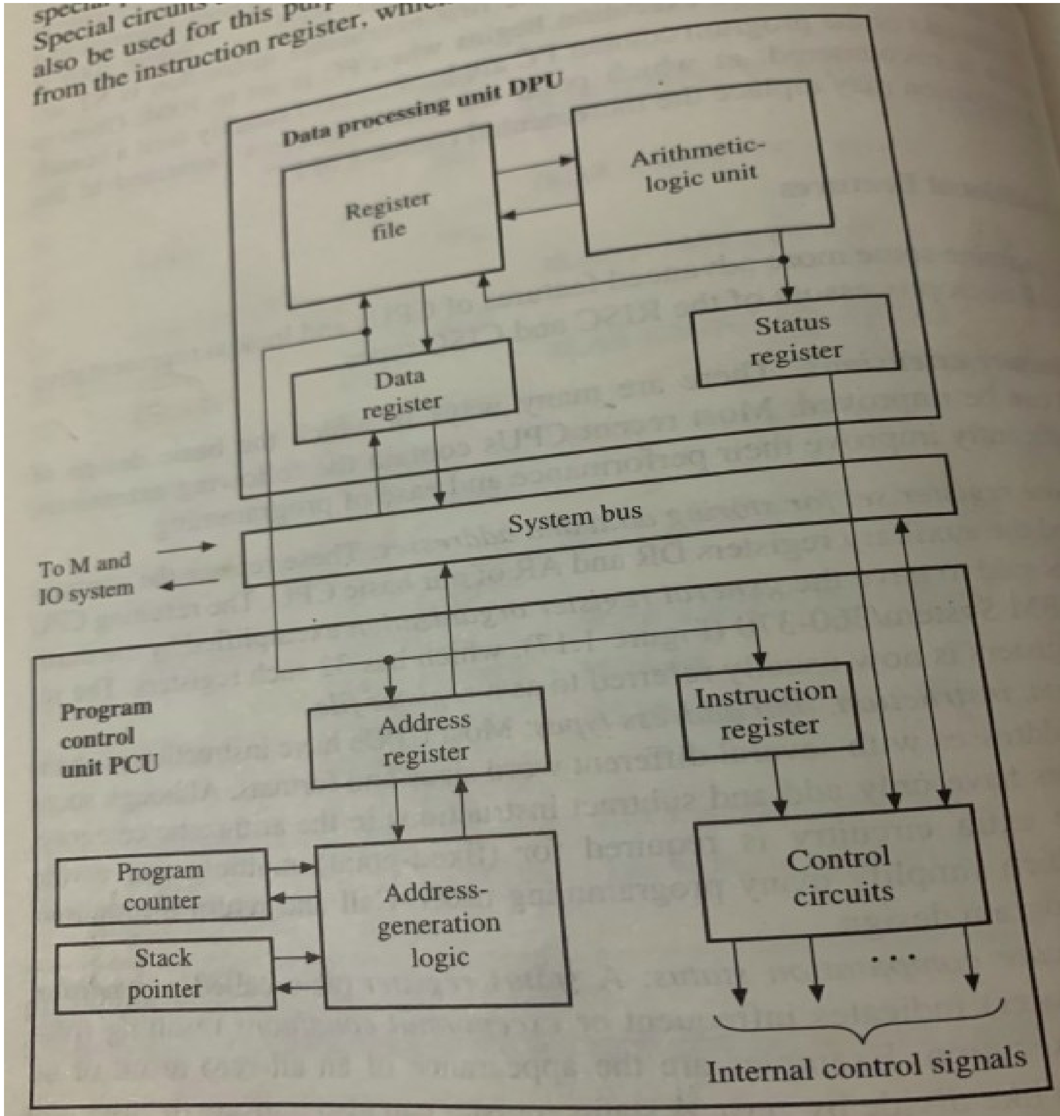


Figure 3.7
A typical CPU with the general register organization.