Q.1) Write short notes on controller and data path of single-purpose processor with block diagram.

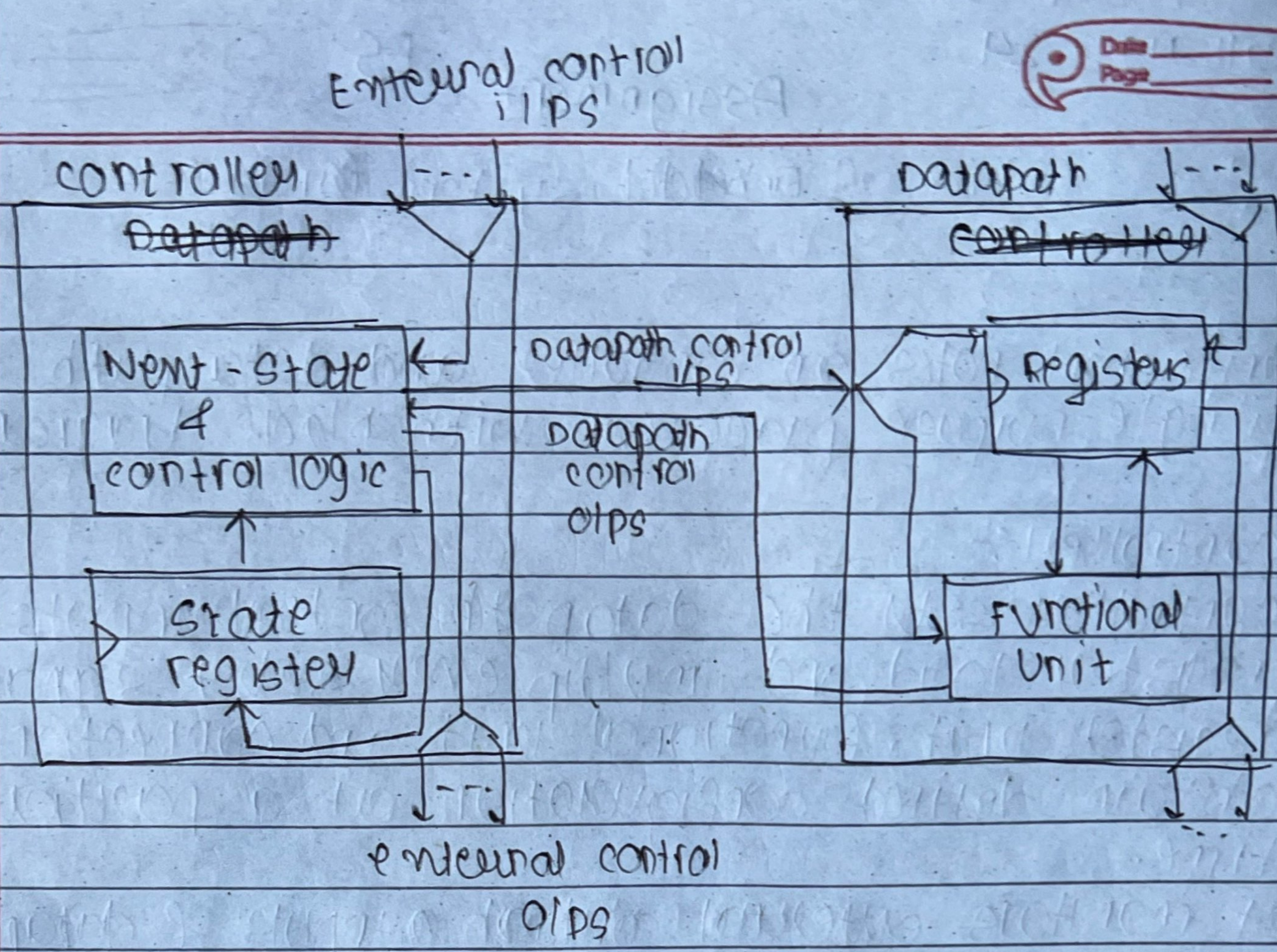
→

Controller :-

- controller sets the datapath control inputs like register load and multiplexer select signals of register unit, functional unit and connection unit to obtain desired configuration at a particular time.
- It monitors external control inputs & datapath control outputs, known as status signal, coming from functional unit and sets external control o/ps as well.

Datapath:

- stores & manipulates system's data.
- consist of register unit, functional unit & connection units like wires & multiplexers.
- It can be configured to read data from particular register, feed that data through functional unit configured to carry out operations like add or shift & store the results back into that reg.
- Ex:- binary no. representing external conditions like speed, temp




Q.2) State complete design steps for single-purpose processor

→

i) Draw block box diagram:-

- It is simple box with external interfaces of a system. It consists of i/p, o/p signals & control signals

ii) Write functionality or program:-

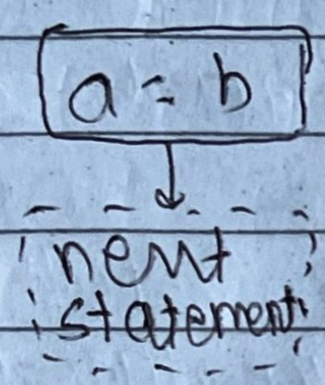
- It is code that provides solution to defined problem.
  - Input signals are assigned to a variable.
  - Temp. variables may be used based on req.
  - Result is assigned to o/p port.

iii) Design FSMD :-

- The code is converted into equivalent Finite State Machine with Data (FSMD).
- In FSMD, templates are used to represent constructs of a program.

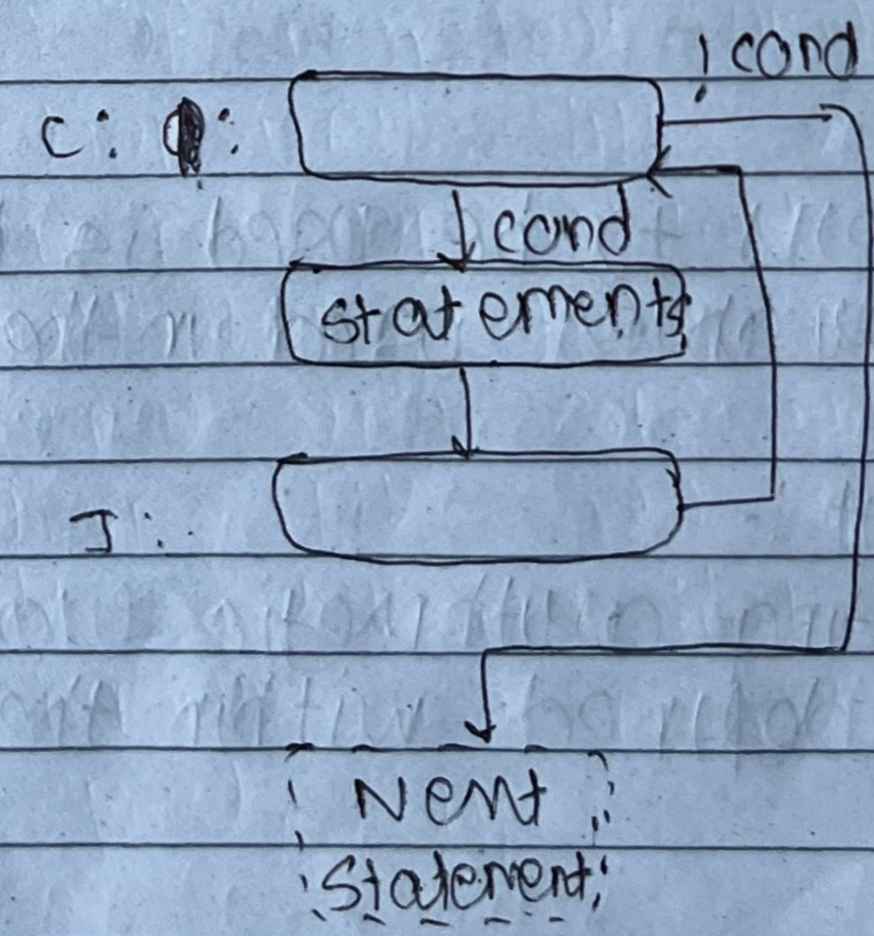
① Assignment statement

```
a = b
next statement
```



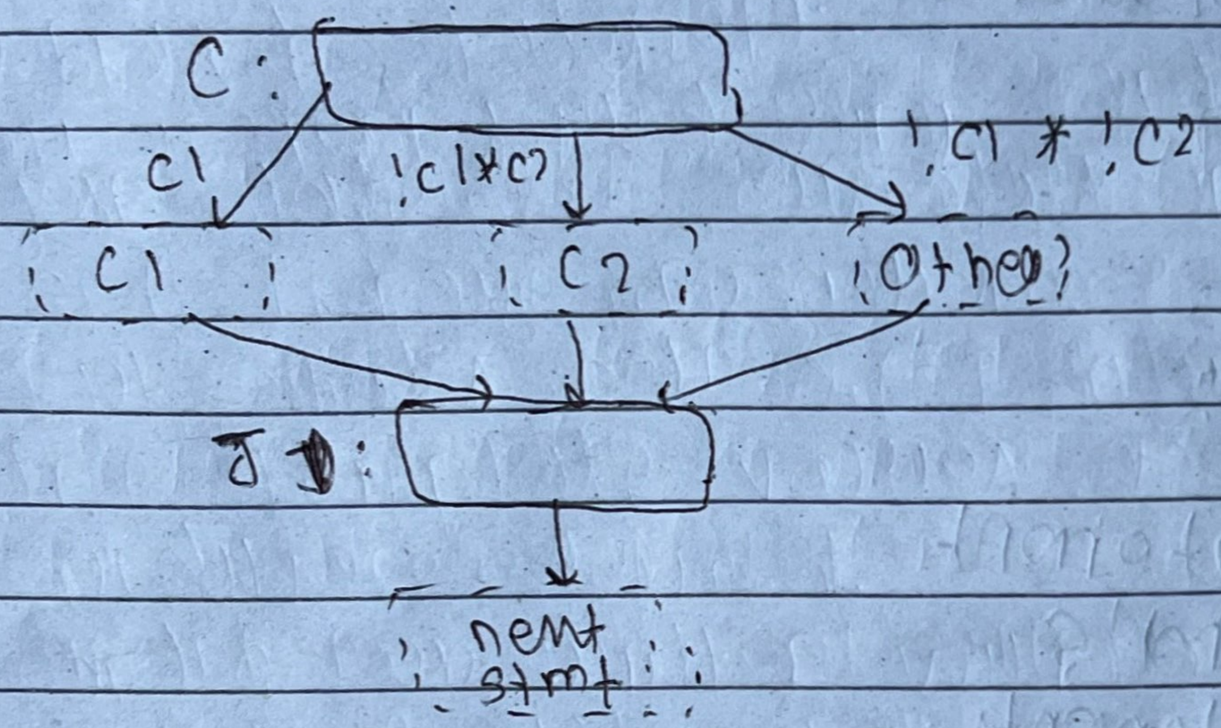

② loop statement

```
while (cond) {
        statements;
}
next statement
```



(3) Branch statement :-

```
if (c1)
        c1 stmts
else if (c2)
        c2 stmts
else
        Other stms
next stmts
```




iv) Build a datapath :-

- It is built based on functionality of the system.

1) Registers :-

- The no. of registers to be used is based on the no. of variables used in the functionality

2) Functional unit :-

- Blocks representing arithmetic & logical operations are defined within the datapath.

3) Connection:-
- The connections betn parts, reg. & functional units are done based on operands used in various assignment & comparison of functionality code.

4) control i/ps & o/ps:
- Input control signals are reqd. by registers & multiplexers. control o/p is produced by logical units of datapath.

v) Develop Finite State Machine (FSM):-
- The states & transitions of FSM are same as that of FSMD.
- However, the complex actions & ~~trans~~ conditions of FSMD are replaced by Boolean expressions using control signals defined within the datapath
- For every register write operation, reg. load signal is asserted & corresponding multiplexer selection line is selected if there are two or more sources for a given register.

Q.3) Write short notes on assignment, branch & loop statements with example.

→

i) Assignment statement:-
- A single state is used with statement representing its action.
- A single arrow is used to connect to next state.

ii) Branch statement:-
- It is represented using condition state C, join state J & few states in betn C & J.
- C & J have no actions & left empty.
- But states betn C & J have actions.
- The template varies according to no. of conditions defined.
- cond'n are written with arrow that connects C-state & other states.
- last states of each branch is connected to J-state.

iii) loop statement:-
- It is represented using cond'n state C, Join state J & states representing loop statements
- cond'n is written alongside arrow joining C state & state of 1st loop statement.
- The last state of loop is connected to J-state which is connected back to C-state.
- complement cond'n is written alongside arrow connecting C state & next statement outside loop.

Q.4) What is optimization? What are the parameters you consider for optimization of single purpose processors.

→ Optimization is the task of making the design metric values the best possible.

- The parameters are:-

i) optimizing original code.

- Analyze program attributes & look for areas of possible improvement.
  - Time & space complexity
  - ~~No.~~ size of variable
  - No. of computation
  - operations used.

Ex:-

```
if (x<y)
    y = y-x;
else
    x = x-y;
}
```

- 9 iterations

```
while (y != 0) {
    r = x % y
    x = y;
    y = r;
}
```

- 3 iterations

Replace '-' with modulo to speed open

ii) optimizing FSMD:

- states in FSMD are assigned with operations from the desired program. This is known as scheduling.

Areas of improvement:-

i) Merge states:-

- States with const. transitions can be eliminated, transition taken is already known
- states with independent operations can be merged

ii) separate states:

- states with complex opers can be broken into smaller states to reduce h/w size.

iii) optimizing datapath:-

- • ~~single fu~~ sharing of functional unit:
- If same oper occurs in diff. states, they can share a single func. unit

• Multi-functional unit:

- ALU can perform variety of operations, it can be shared among operations occuring in diff. states.

iv) Optimizing FSM:-

• State Encoding

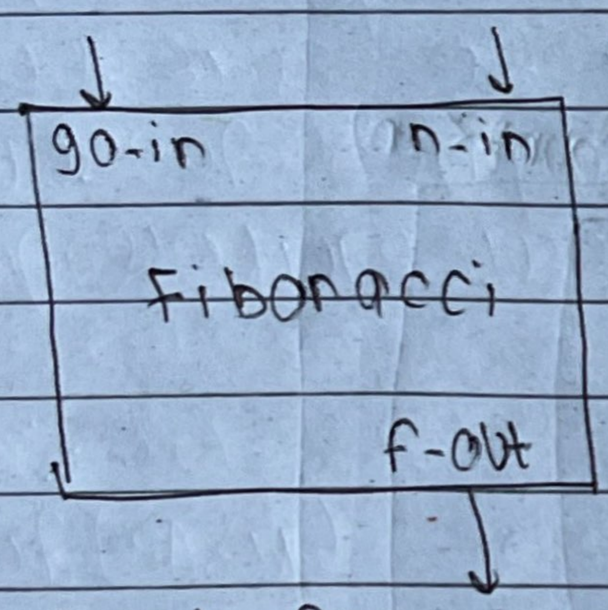
- task of assigning unique bit pattern to each state in FSM.
- size of state reg. & combinational logic vary.

0 1 1 2 1 1 2 1 2 3 2

• State minimization :-
- task of merging equivalent state into a single state.

6
a.) Design a single purpose processor that outputs Fibonacci numbers upto 'n' places. Start with a function computing the desired result, translate it into state diagram and sketch a probable data path.

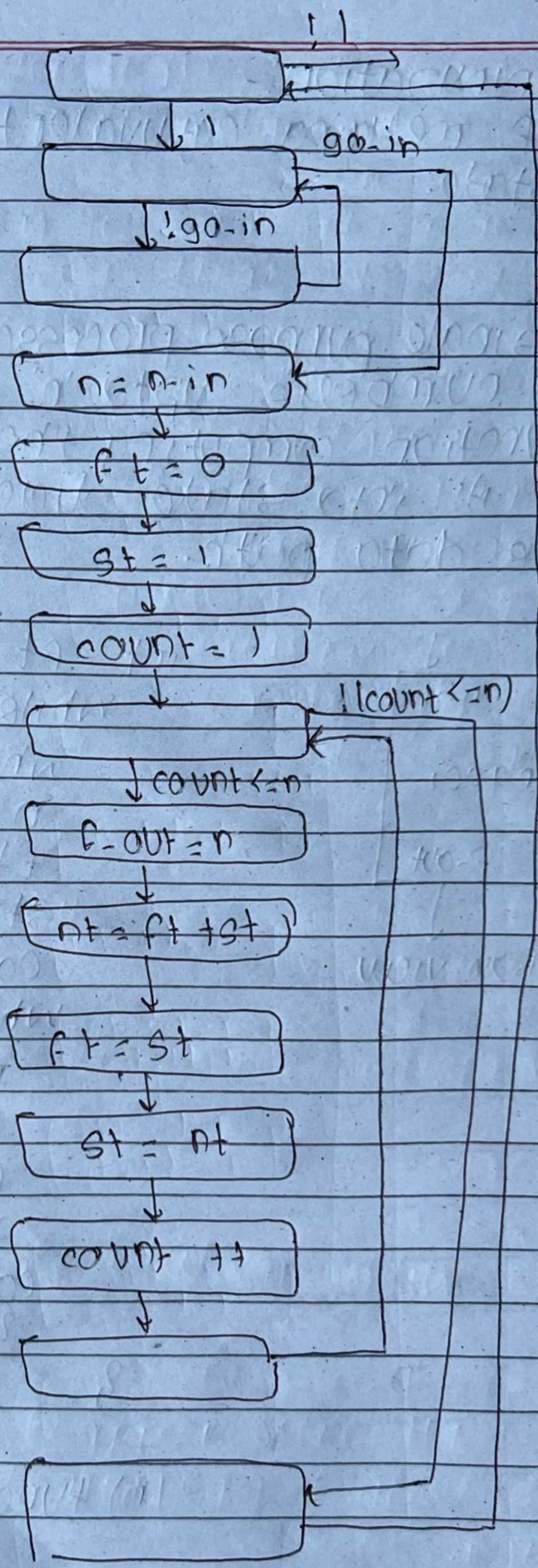
→



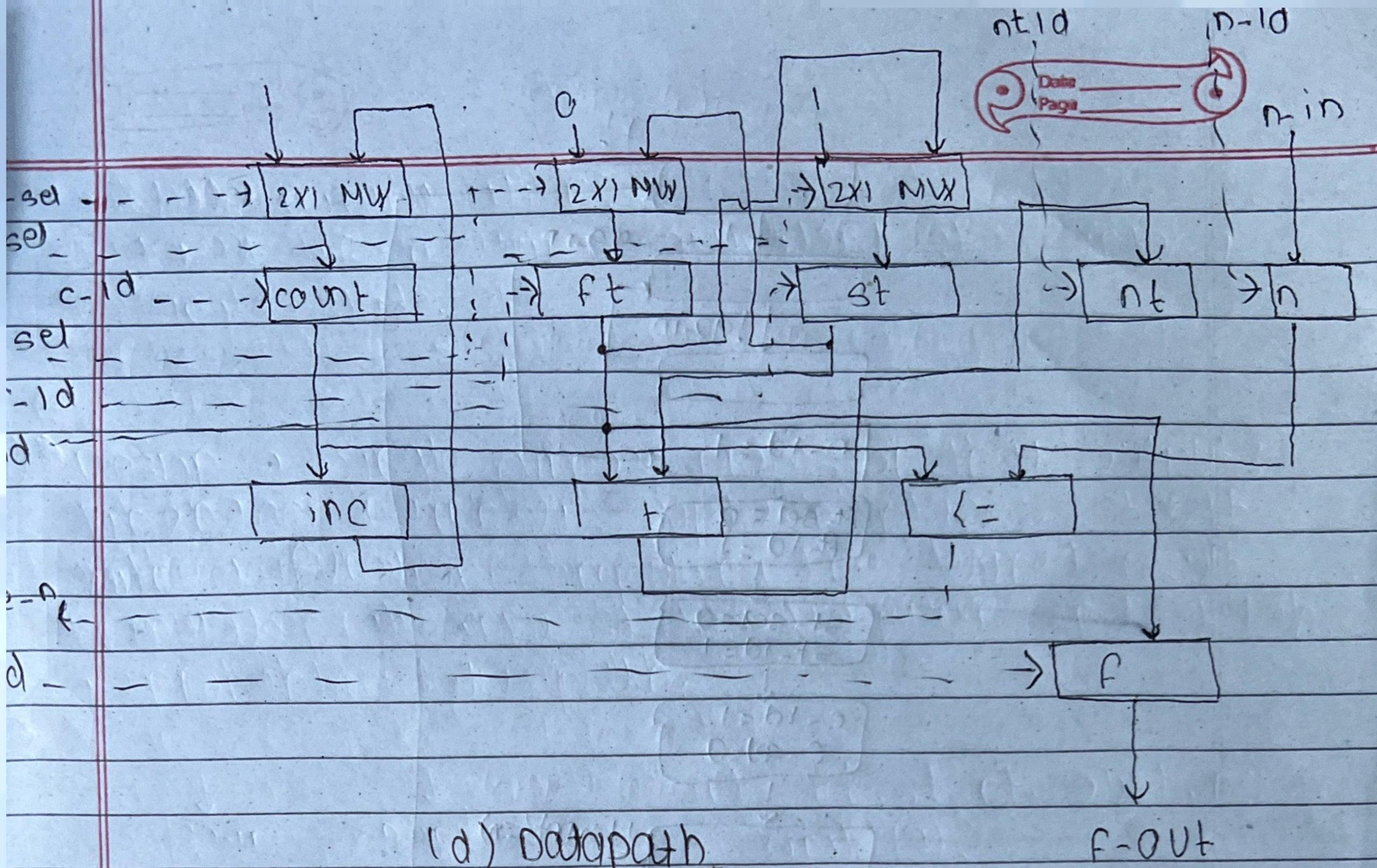

a) Black Box view

```
int ft, st, nt, count, n;
while(1){
  while(!go-in);
  n = n-in;
  ft = 0;
  st = 1;
  count = 1;
  while(count <= n){
    f-out = ft;
    nt = ft + st;
    ft = st;
    st = nt;
    count ++;
  }
}
```

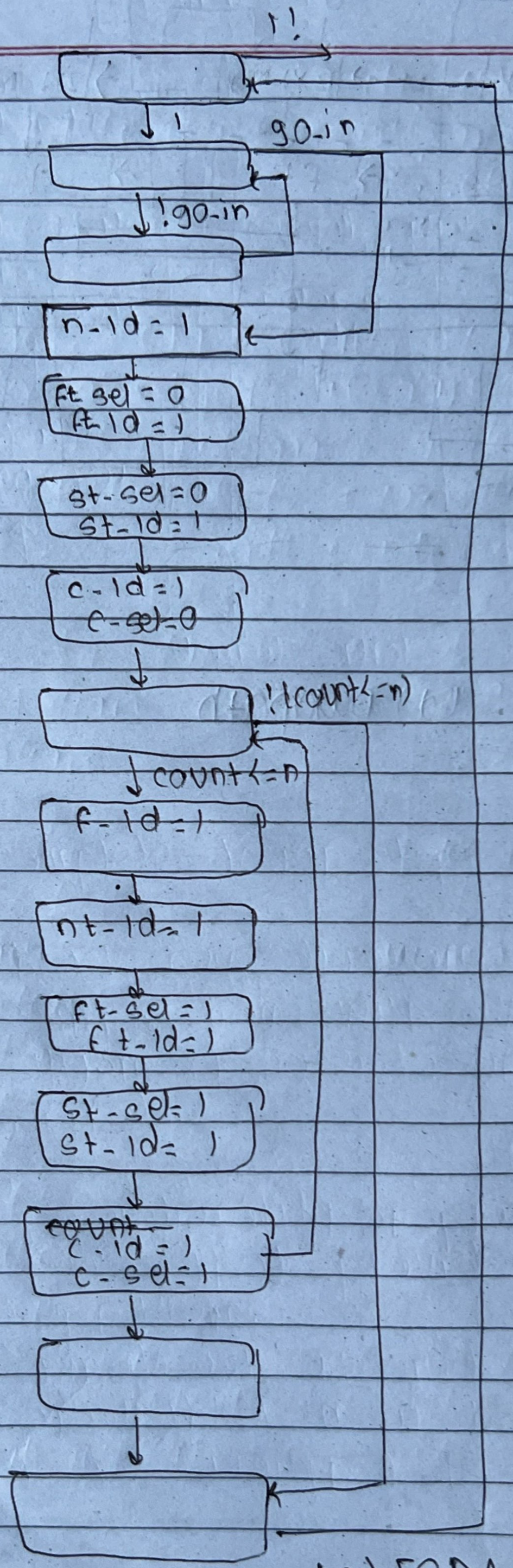
(b) Functionality


go_in
!go_in
n = n_in
ft = 0
st = 1
count = 1
!(count <= n)
count <= n
f_out = n
nt = ft + st
ft = st
st = nt
count ++

1c) FSMD


nt_ld
n_ld
n_in
1
0
1
2X1 MUX
2X1 MUX
2X1 MUX
count
ft
st
nt
n
inc
+
<=
f
f_out

(d) Datapath


!!
go_in
!go_in
n_ld = 1
ft_sel = 0
ft_ld = 1
st_sel = 0
st_ld = 1
c_ld = 1
c_sel = 0
!(count<=n)
count<=n
f_ld = 1
nt_ld = 1
ft_sel = 1
ft_ld = 1
st_sel = 1
st_ld = 1
~~count~~
c_ld = 1
c_sel = 1

(e) FSM

Q.4) ~~What is optimization? What are the parameters you consider for optimization of SPP?~~

→

Q.5) Define data path and controller of a general purpose processor (GPP). Explain ASIP with its types.

→ Data path contains circuitry for transforming data & temporary data storage.
controller sequences through states & generates control signals to read instr^n into the IR & control flow of data betn reg, ALU of datapath & memory.

ASIP are specific to the particular appl^n domain. They can be programmed based on req. of particular area which makes it more flexible.
Ex:- video processing, network processing, etc.

Types:-

① Microcontroller:-
For embedded control appl^ns:-
- Reading sensors, setting actuators
- Mostly dealing with bits: data is present but not in huge amount.
- Ex:- VCR

Microcontroller features :-
- on chip peripherals like timers, ADCs, serial commn devices.
- on chip program & data memory.
- Direct access of programmer to many chip's pins.
- specialized instrns for low-level instrns & bit manipulation.

(2) DSP :-
- For signal processing applicns :-
- large amount of digitized data
- Data transfr must be applied fast
- Eg:- cell-phone voice filter

DSP features :-
- some DSPs allow concurrent execution of funcns which boost the performance of system.
- contains peripherals like ADC, DAC, PWM, timers, counters, DMA controllers etc.
- frequently used arithmetic funcns are implemented using h/w which results in faster execution compared to s/w implementn.

(3) Less-General ASIP :
- They are designed to perform domain specific processing allowing some degree of programmability.
- Eg:- Processors designed for networking h/w

| | | |
|---|---|---|
| v) | does not support rotating priority | v) supports rotating priority. |
| vi) | simple | vi) complex |
| vii) | small no. of devices | vii) handle large no. of devices |

Q.1) Define pipelining and show 6 stage pipeline concepts. Explain DSP with characteristics & advantages.

→ Pipelining is the arrangement of h/w elements of CPU such that its overall performance is increased by simultaneous execution of more than one instrn.

6 stages:-

i) Instrn fetch:-
- CPU retrieves instrn from memory

ii) Instrn Decode:
- CPU decodes instrn to determine the opern to be performed

iii) calculate operands:
- CPU calculates address of operands read by the instrn

iv) operand Fetch

v) Execute Instrn

vi) Write Operand

DSP:-

- Digital signal processor (DSP) ~~contain~~ are processors which are specific to appl^ns that process large amount of data.

Characteristics:-

i) concurrent execution
ii) arithmetic func
iii) peripherals
iv) contains multipliers, memory blocks, reg. files etc.

Advantages:-

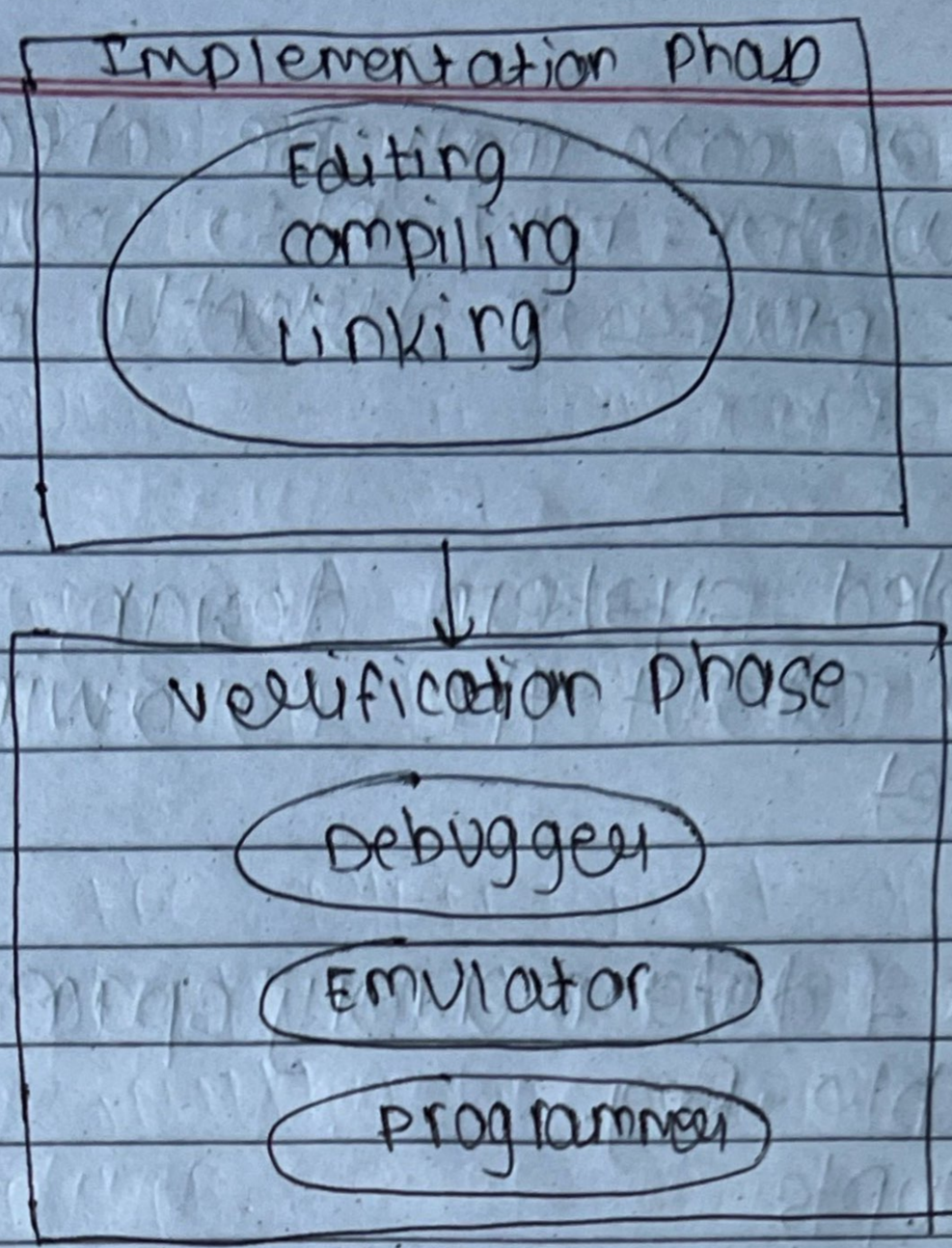
i) provides flexibility
ii) less susceptible
iii) Improves performance
iv) supports complex operations

Q.8) Explain the design flow of embedded s/w development. Explain in brief about programmer's view for general purpose processor.

→ In case of embedded system design, the target & development processors are diff. for all system

- The IDE tools for various processors are available for implementation phase.

9.) 8




a) Implementation phase:-
- source code is written by editor & is compiled/ assembed using compiler/assembler.
- linker ~~converts~~ combines all reqd. files into a single executable file.
- Development processor use cross-assembler or cross-compiler.

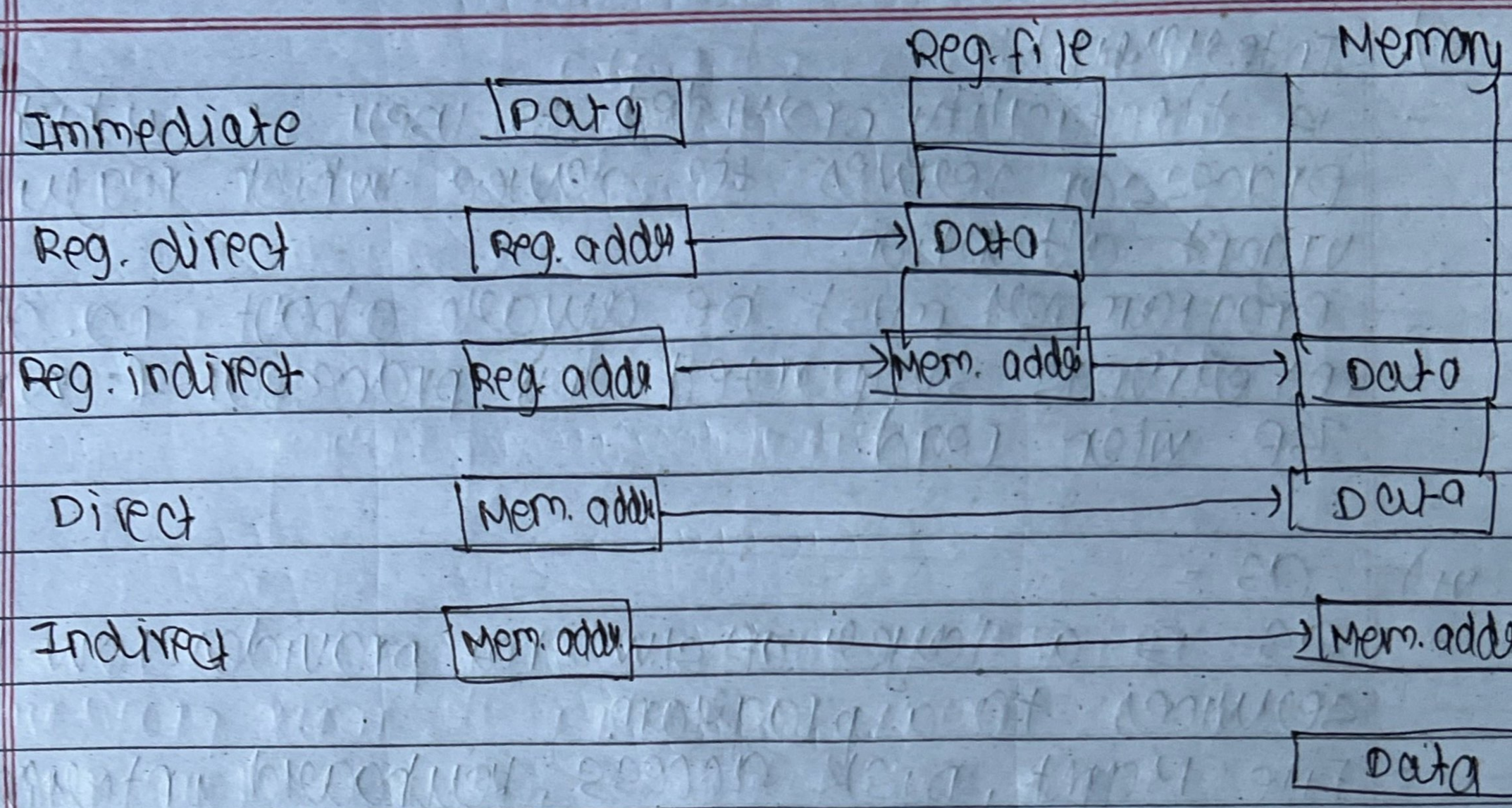
b) verification phase:-
- Embedded system works in conjunction with other components as well as real-time env so debugging a program requires control over time & env.
- Based on req. & availability, debugger, emulator & programmer can be used for verification.
- code may be loaded on dev. processor by

debugger or code may be checked by loading into emulator h/w. Also code can be loaded to target processor directly by programmer

In embedded system design, programmer must be aware about the following:-

i) Instr^n set
ii) OS
iii) Program & data memory space
iv) Available Reg.
v) Interrupts
vi) I/O Facility

i) Instr^n set:-
- Defines the legal set of instr^ns that can be carried out by the processor.
- Programmer must be aware about what instr^ns can be executed.
- 2 levels:-
  - Assembly level
  - Structured languages




ii) Program & Data memory space
- Programmer must be aware about available size for program & data.
- For eg. in ~~50~~ micro-controllers, on-chip memory for program & data are fixed.

iii) Available Reg:-
- programmers must be aware about no. of reg. available for general purpose & specific purpose

iv) I/O facility:-
- Each processor facilitates programmer with I/O pins to communicate with external devices.
- programmer must be aware about no. of pins & their func.

v) Interrupt:
- is the facility provided to user in which the processor serves the device which requires urgent attention.
- programmer must be aware about no. of interrupts supported by processor & write ISR when reqd.

vi) OS :-
- OS is a layer of s/w that provides low-level services to a program.
- File Mgmt., Disk access, keyboard interfacing.
- Scheduling multiple programs for execution.
- programs make sys. call to OS.

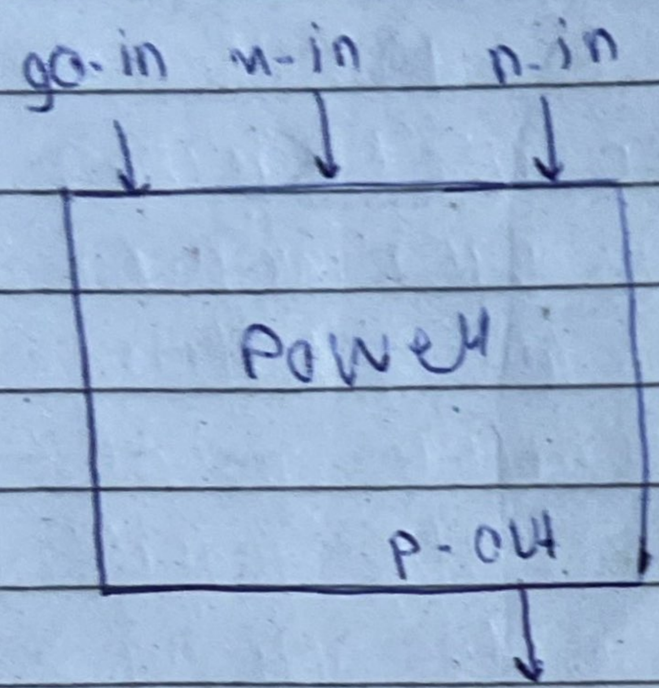
~~Q.10) How interrupts interface with microprocessor? Explain it. Briefly explain operation of peripheral to memory transfer with DMA.~~

→

Q.9) Design a SPP to determine value of m to the power n. Start the design from the function computing the desired result, FSMD datapath and the controller.

→

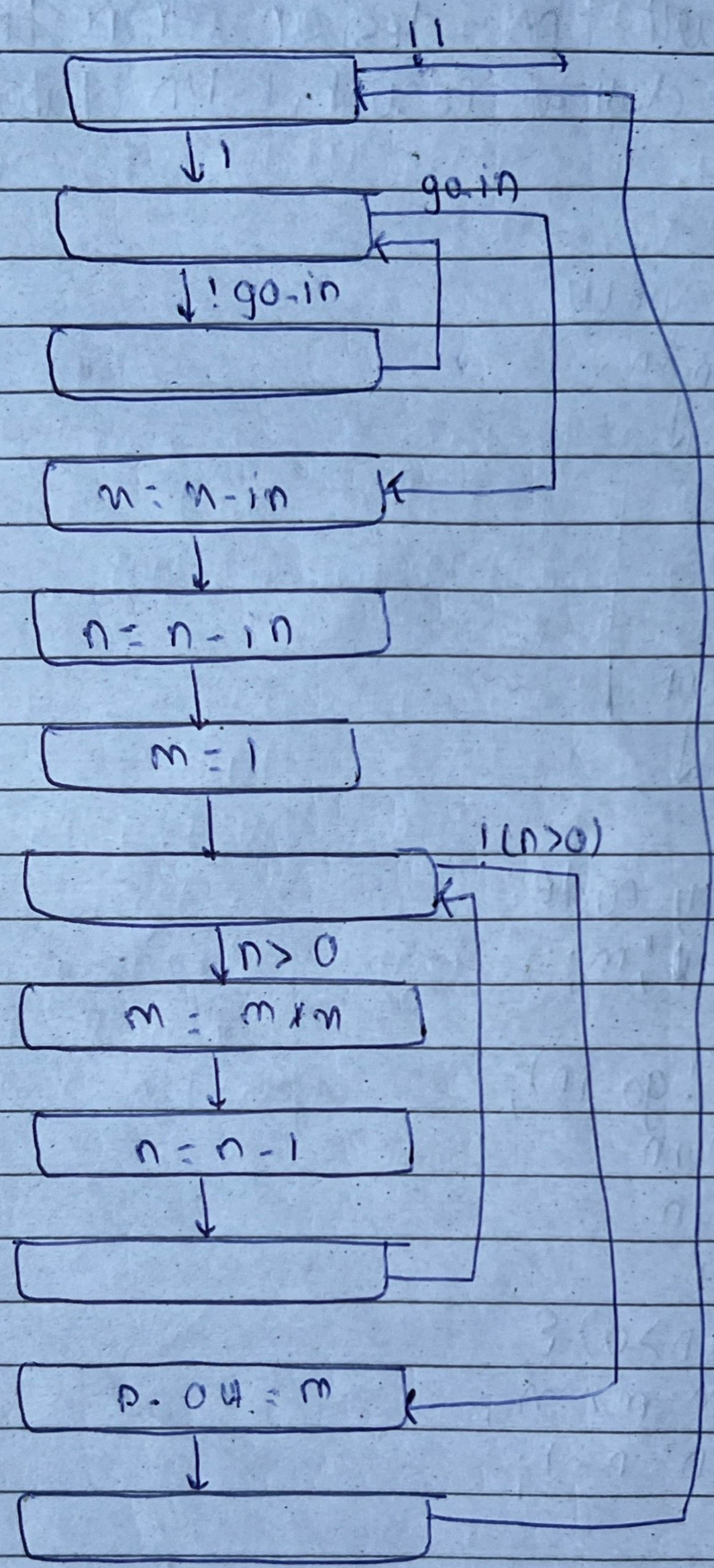
a) Block box view



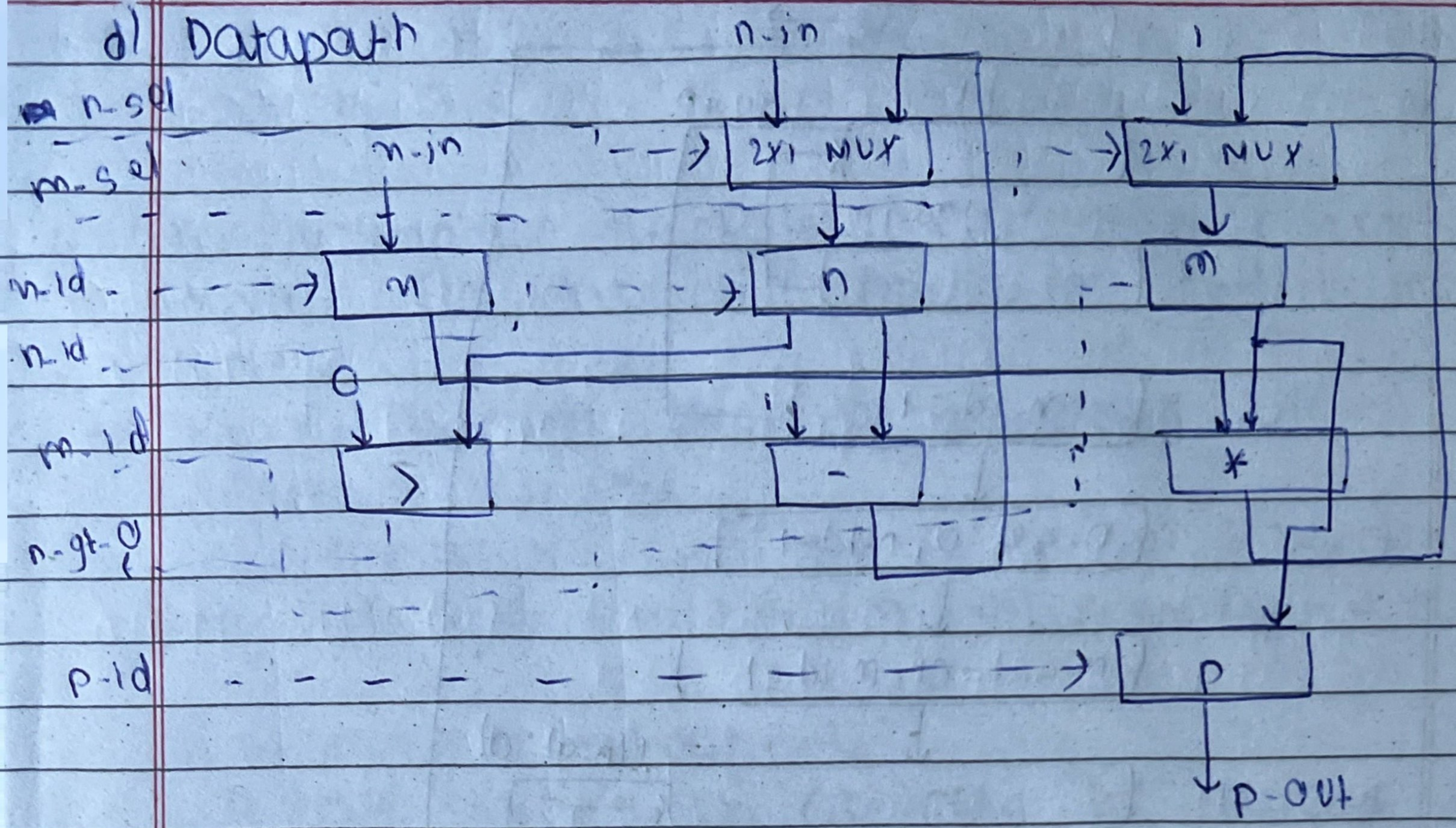

b) Functionality code:

```
int m, n, p, m;
while (1){
    while (!go-in);
    m = m-in
    n = n-in
    m = 1
    while (n>0) {
        m = m * m;
        n = n - 1;
    }
    p-out = m;
}
```

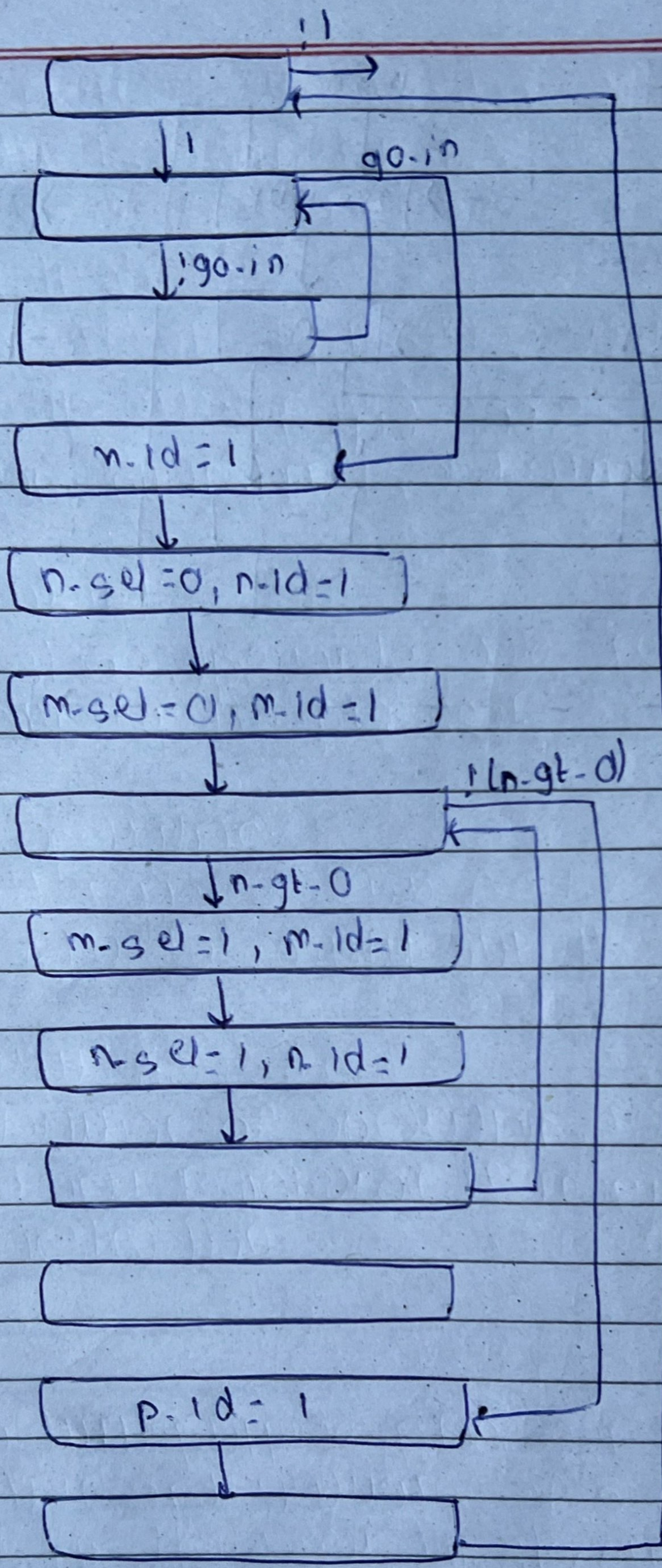
c) FSMD:



d) Datapath




e) FSM:-


!1
1
go-in
!go-in
n-id=1
n-sel=0, n-id=1
m-sel=0, m-id=1
!(n-gt-0)
n-gt-0
m-sel=1, m-id=1
n-sel=1, n-id=1
p-id=1

Q. 10)

→ Same as Q. 6

Q. 11) Briefly explain the criterion for selecting processor? Explain the datapath operation and its instruction cycles.

→ The criterion for selecting processor are:

i) Technical aspects:

- Selection of processor must be done based on required speed within limited power, size and cost

ii) Non-technical aspects:

- Before selecting µp, one must be aware of development environment, prior expertise of processor licensing arrangements and so on

Datapath consists of the circuitry for transforming data and for temporary data storage. It contains ALU which manipulates data through various operations like addition, subtraction, logical AND, OR etc. It also consists registers to store temporary data and different status generated by operations.

The instruction cycle is divided into the following phases:-

i) Fetch instruction:

- The next instrn to be executed is loaded into

IR from memory. The address of memory where instr resides is given by PC.

ii) Decode instruction:
- Instruction in IR represents various operations based on op-code. In this stage, the operation to be done by instrn is determined.

iii) Fetch operand:
- In this stage, operands are retrieved from the memory by the processor.

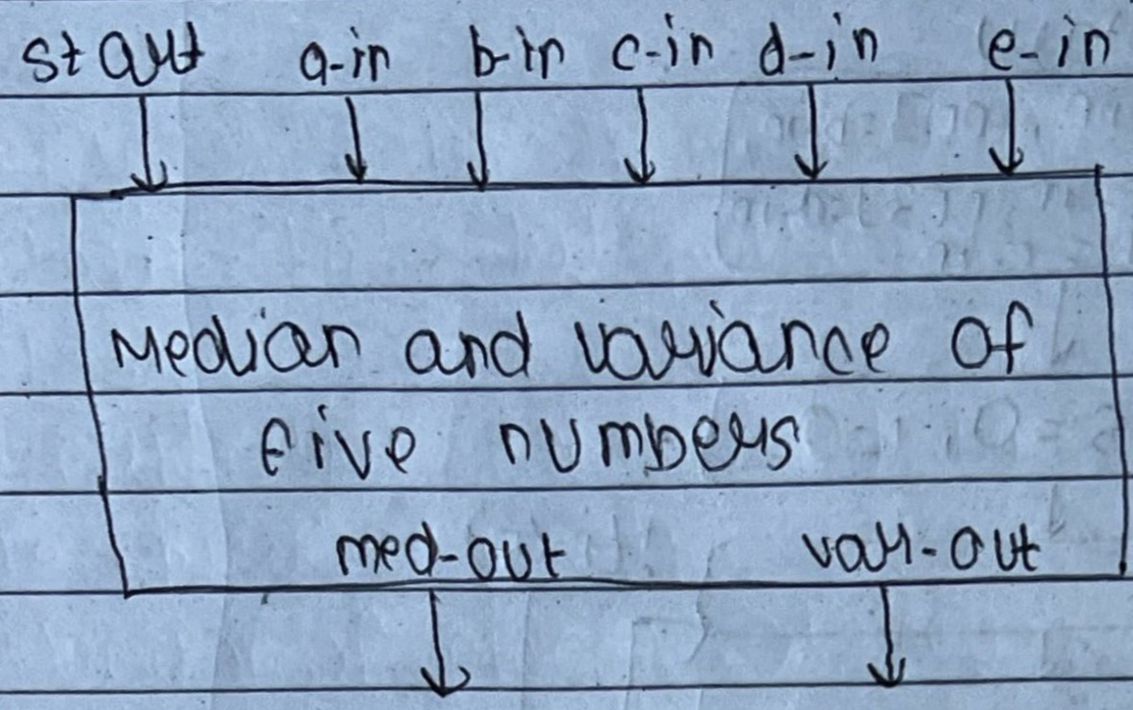
iv) Execute operation:
- The loaded registers are fed to the inputs of the ALU to carry out the operation.

v) Store results:
- After execution of operation, the final data is loaded into register or memory as defined by the instruction

Q.12) Design a dual purpose processor that calculates the median and variance of 5 numbers entered by the user, by showing the algorithm, FSMD, FSM, datapath and controller design.

→

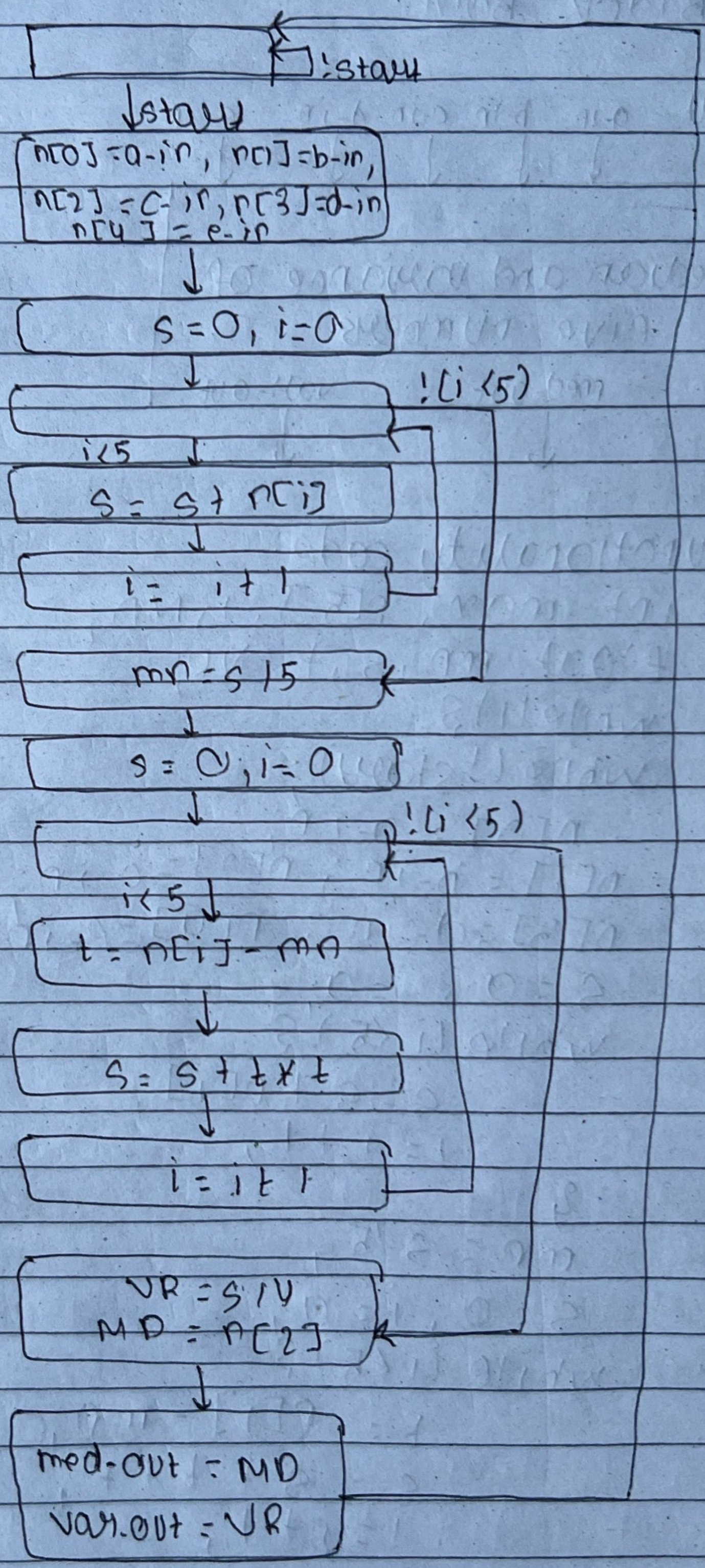
a) Black Box



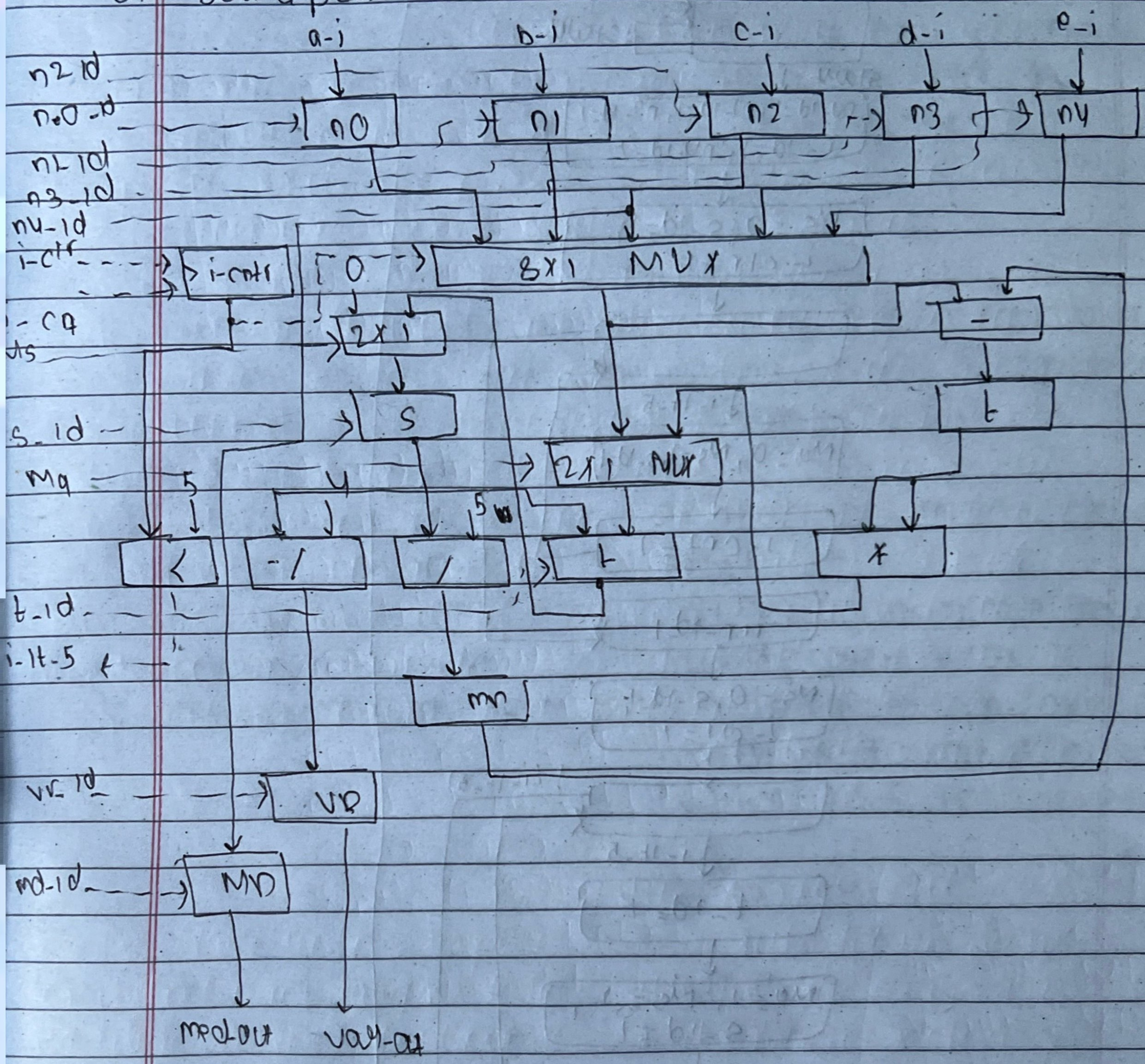

b) Functionality code:

```
int mon, n[5], i, MD;
float mn, s, t, VR;
while(1){
while(!start);
  n[0] = a-in;
n[1] = b-in; n[2] = c-in.
n[3] = d-in; n[4] = e-in;
s = 0, i = 0;
while(i<5){
        s = s + n[i];
        i = i + 1;
}
mn = s/5;
s = 0, i = 0;
while(i<5){
        t = n[i] - mn;
        s = s + t * t;
        i = i + 1;
}
VR = s/4;
MD = n[2];
m-out = MD;
v-out = VR;
```

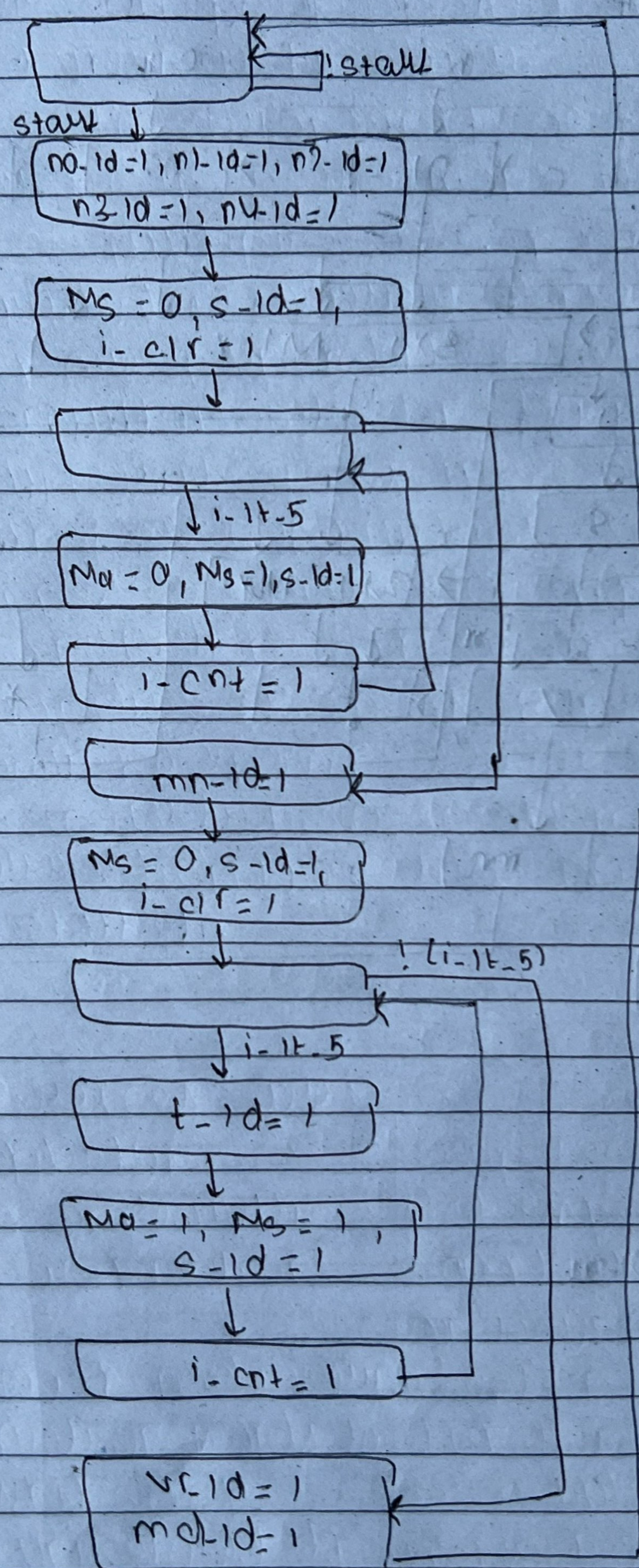
c) FSMD:


start
start
n[0] = a-in , n[1] = b-in ,
n[2] = c-in , n[3] = d-in
n[4] = e-in
s = 0, i = 0
!(i < 5)
i < 5
s = s + n[i]
i = i + 1
mn = s / 5
s = 0, i = 0
!(i < 5)
i < 5
t = n[i] - mn
s = s + t * t
i = i + 1
VR = s / 4
MD = n[2]
med-out = MD
var-out = VR

d) Datapath:


a_i
b_i
c_i
d_i
e_i
n2_id
n0_id
n1_id
n3_id
n4_id
i_ctr
n0
n1
n2
n3
n4
i_cntr
8x1 MUX
-
2x1
s
t
2x1 MUX
5
4
5
<
-/
/
t
*
t_id
i_lt_5
mn
vr_id
VR
md_id
MD
mdout
vout

e) FSM:


!stall
stall
n0_ld=1, n1_ld=1, n2_ld=1
n3_ld=1, n4_ld=1
Ms=0, s_ld=1,
i_clr=1
i_lt_5
Ma=0, Ms=1, s_ld=1
i_cnt=1
min_ld=1
Ms=0, s_ld=1,
i_clr=1
!(i_lt_5)
i_lt_5
t_ld=1
Ma=1, Ms=1,
s_ld=1
i_cnt=1
vr_ld=1
md_ld=1

Q.13) Differentiate between application specific instruction set processor and general purpose processor. Also discuss on issues related to selection of a particular processor.

→ The differences are:-

| ASIP | GPP |
|---|---|
| i) Designed for a specific or narrow range of applications | i) Intended to handle a wide variety of applications |
| ii) limited flexibility due to its specialization | ii) High flexibility |
| iii) Highly optimized for a specific set of tasks | iii) less optimized for specific tasks. |
| iv) Designing ASJD is time-consuming | iv) Readily available reducing development time. |
| v) Development cost is high | v) Mass production & availability contribute to cost efficiency |
| vi) Ex:- DSP | vi) Ex:- x86, ARM processors |

Q.IV) Explain the concept of pipelining. Differentiate super scalar and VLIW architecture.

→ Pipelining is the arrangement of h/w components of CPU to increase the performance by simultaneous execution of multiple tasks. In pipeline, the given task is divided into various stages and multiple stages which are independent of each other are executed simultaneously. For efficient instruction pipeline, different stages must be of almost same length & each instruction must require same number of cycles to complete its execution. Branching instructions can be an obstacle for efficient pipeline as next instruction to be executed will be known after execution stage of branch instruction. This problem can be addressed using various techniques like branch prediction, pipeline stall etc.

The differences are:

| | VLIW architecture | Super-scalar architecture |
|---|---|---|
| i) | It receives single multi-operation instruction | i) It expect for a traditional sequential flow of instr but can issue multiple instr's |
| ii) | VLIW approach needs very long instruction words to specify what each execution unit should do | ii) It receives sequential streams then it is decoded and the issues unit will issue multiple instructions for multiple execution units. |

| | | |
|---|---|---|
| iii) | It expect dependency free-code | iii) It does not expect dependency free code ~~not~~ deals with dependencies using special h/w |
| iv) | VLIW is less complex | iv) Superscalar processors with same degree of parallelism are more complex than VLIW architecture. |
| v) | Used for static scheduling | v) Used for dynamic scheduling. |

Q.15) How ASIP serve as a compromise between GPP and SPP? Briefly explain the programmers' considerations on GPP.

→