INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE CENTRATOM TRIESTE

H4.SMR. 403/30

FIFTH COLLEGE ON MICROPROCESSORS: TECHNOLOGY AND APPLICATIONS IN PHYSICS

2 - 27 October 1989

Real Time Systems

L. ZANELLO
Universita' "La Sapienza", Dept. of Physics, Roma, Italy

A QUOTATION :

" IT OFTEN APPEARS THAT THE WORLD MAY BE PARTITIONED INTO TWO GROUPS : THOSE WHO TELL OTHERS HOW TO DESIGN SYSTEMS, AND THOSE WHO ACTUALLY DO IT ".

WORKING PROGRAM

- DEFINITION OF REAL-TIME SYSTEM
  .(r.t.s. vs computational)
- CLASSIFICATIONS, AND WHAT WE ARE NOT GOING TO TALK ABOUT.
- THE THREE COMPONENTS OF A R.T.S. :
  - process control
  - monitoring and logging
  - data acquisition
- VERY GENERAL STRUCTURE OF A R.T.S.
  - the hardware
  - the control software
  - links to the external world
  - user interaction
  - debugging
  - upgrading and maintenance
- EXAMPLES 1, 2, 3 :
  - computer-controlled hot strip mill
  - X-rays industrial tomography
  - a detector for high energy physics

ORGANIZATION OF A R.T.S.

OF A TRADITIONAL (UNIPROCESSOR)

COMPUTING SYSTEM :

- LANGUAGES
- HARDWARE CONTROL
- MEMORY - MAPPED I/O
- DATA BUFFERS
- INTERRUPTS
  AND HOW TO HANDLE THEM
- STACK - REENTRANT CODE
- PRODUCERS AND CONSUMERS
- THE SHARED RESOURCES
  CRITICAL AREAS AND
  HOW TO HANDLE THEM
- DATA BASES FOR MONITORING
- MULTI-TASK ORGANIZATION;
  CONCURRENCY
- DEBUGGING
- HOW TO MAKE THE R.T.S.
  USER-FRIENDLY
  (SMALL TALK)
- HOW TO MAKE THE R.T.S.
  GROW OLD, GRACEFULLY.

MANY -
MICROPROCESSOR - BASED R.T.S. :

- MULTIPROCESSING VS. MULTITASKING
- HARDWARE / SOFTWARE SPLIT
- MASTER / SLAVE ORGANIZATION
- SYNCHRONIZATION
- LANGUAGES
- SHARED RESOURCES
- COMMUNICATION
- DEBUGGING (!!)
- MAINTENANCE
- FORMAL TREATMENT OF THE CONCURRENCY
  PROBLEM

CONCLUSIONS

CLASSIFICATIONS AND WHAT WE AR NOT GOING TO TALK ABOUT (AND WHY).

- WHAT IS CLASSIFIED AS "REAL TIME" IN THE LITERATURE:

EVERY APPLICATION WHERE THE COMPUTER IS BOUND TO ANSWER TO A GIVEN REQUEST WITHIN A CERTAIN TIME (WHICH CAN μsec, msec, sec..).

THIS INCLUDES TWO WIDELY DIFFERENT TYPES OF APPLICATIONS, WHICH SHARE A FEW CONSTRAINTS BUT ARE CHARACTERIZED BY VERY DIFFERENT KINDS OF COMPUTERS AND SOFTWARE TECNIQUES:

a) AIRLINES RESERVATIONS, STOCK EXCH. CONTROL, BANK ACCOUNT HANDLING .....

b) INTRUMENTATION CONTROL, EQUIPMENT MON.TOR, FAST DATA ACQUISITION, ROBOTS .....

WHAT THEY HAVE IN COMMON: NECESSITY OF

- (ALMOST) SIMULTANEOUS HANDLING OF REQUESTS

AND

- ACCESS TO SHARED RESOURCES

BUT:

WHILE SYSTEMS OF TYPE) ARE NORMALLY IMPLEMENTED ON BIG MACHINES, WITH POWERFUL OPERATING SYSTEMS, VERY LARGE AMOUNTS OF MASS STORAGE, SOPHISTICATED NETWORK INTERCONNECTIONS AND A STRONG SEPARATION BETWEEN SYSTEM AND APPLICATION, SYSTEMS b) CAN BE BASED ON VERY SMALL AND CHEAP COMPUTERS (E.G. 1 MICROCOMPUTER), ARE BASICALLY LOCAL CONTROL SYSTEMS AND DEMAND A STRONG INTEGRATION BETWEEN THE "SYSTEM" AND THE "APPLICATION" (MANY TIMES WE CANNOT EVEN MAKE THIS SEPARATION)!! IN MANY INSTANCES YOU MAY HAVE TO WORK ON THE "BARE" MACHINE (NO OP. SYST. OR "KERNEL MODE" ETC.).

WE ARE GOING TO DISCUSS TYPE b), WHERE MICROCOMPUTERS AND MICROPROCESSORS FIND THEIR NATURAL APPLICATION FIELD:

- IN PARTICULAR, WHAT IS INTERESTING AND CHALLENGING (AND NEW) IS THAT IN THESE SYSTEM, THERE IS NOW A LARGE SPECTRUM OF CHOICES AT THE DESIGN LEVEL, WHICH INCLUDES THE COMPUTER, WHILE PREVIOUSLY, WITH MINICOMPUTERS, THE CHOICE WAS LIMITED TO "FACTORY" AND "SIZE".

2

DEFINITION OF REAL-TIME SYSTEM.

1) AN APPLICATION, UNDER COMPUTER CONTROL, IN WHICH TIME IS A CRITICAL PARAMETER

AND, FOR US:

2) WHERE THE INTERACTION WITH SOME SORT OF EQUIPMENT IS NECESSARY.

NEW, FASHIONABLE TERM: "EMBEDDED SYSTEM"

DO NOT INTERPRET 1) IN TOO WIDE A SENSE OR ELSE

EVERYTHING IS REAL-TIME

THE BASIC SUBUNIT OF OUR R.T.S. IS THE

TASK

WE SHALL DEFINE IT AS A SEQUENTIAL PROGRAM PERFORMING A WELL DEFINED FUNCTION.

ADDING TO A TASK THE SET OF STATE INFORMATION NECESSARY TO BE ABLE TO INTERRUPT AND RESTART IT AT ANY POINT, WE GET A

PROCESS

A REAL-TIME SYSTEM IS ALWAYS A MULTITASK SYSTEM AND, WITH RESPECT TO OTHER TYPES OF SYSTEMS, IT TENDS TO SHOW A HIGH DEGREE OF INTERACTION AMONG THESE TASKS (PROCESSES).

PROCESSES ARE INDEPENDENT IF THEY DO NOT COOPERATE NOR COMPETE WITH EACH OTHER

PROCESSES ARE CONCURRENT IF THEY INTERACT BY COOPERATING, SHARE AND COMPETE FOR RESOURCES.

IN NON R.T.S. MULTIPROCESSING AND MULTITASKING ARE USED TO INCREMENT THE TOTAL SYSTEM THROUGHPUT.

IN R.T.S. CONCURRENCY IS INHERENT AND WE ARE READY TO SACRIFICE THROUGHPUT IN EXCHANGE FOR RESPONSE TIME: THE COMPUTER(S) CAN STAY IDLE PART OF THE TIME, PROVIDED THEY ARE READY TO GIVE THE PROPER ATTENTION AT THE PROPER TIME.

SO, WHAT WE ARE INTERESTED IN DISCUSSING IS THE ORGANIZATION OF SYSTEMS WHERE SOME HARDWARE IS TO BE ACTIVELY CONTROLLED OR MONITORED, WHERE DATA (SOMETIMES LARGE AMOUNTS OF IT) MUST BE COLLECTED IN A SHORT

THE HARDWARE IS SUBJECT TO FREQUENT MODIFICATIONS AND UPGRADING, AND WHERE LOW-COST, FLEXIBLE, DISTRIBUTED, EASY-TO-INTERFACE AND CHEAP COMPUTING TOOLS ARE NECESSARY.

⊢

WE ARE NOW TO DISCUSS:

THE THREE BASIC COMPONENTS OF A R.T.S.

a) PROCESS CONTROL : THE EQUIPMENT (MOTOR, FURNACE , TOOL, MEDICAL INSTRUMENTS , HIGH VOLTAGE SUPPLY , ROBOT ...) IS REGULATED AND CONTROLLED BY SETTINGS OF VALVES, SWITCHES, CONTROL VOLTAGES ETC ...
A/D CONVERTERS BRING TO THE COMPUTER NUMERIC VALUES DESCRIBING THE CURRENT SYSTEM STATUS . THESE ARE COMPARED WITH PRE-STORED OPTIMAL VALUES AND TOLERANCES .
CORRECTIONS ARE COMPUTED AND NEW VALUES ARE D/A CONVERTED TO MODIFY THE SETTING POINTS.
THIS IS ALWAYS ACCOMPANIED BY LOGGING ACTIVITY , I.E. RECORDING OF TIME-DEPENDENCE OF SYSTEM PARAMETERS AND BY ALARMS AND AUTOMATIC CUT-OFFS FOR EMERGENCIES.
THE CRITICAL PARAMETER HERE IS RELIABILITY

b) MONITORING : INCOMING DATA ARE SAMPLED , ANALYSED AND SHOWN TO A USER, WITH THE PURPOSE OF CHECKING THAT THE APPARATUS IS WORKING PROPERLY AND THAT THE DATA ACQUISITION PARAMETERS (EVENT SELECTION, CUTS ETC) ARE CORRECTLY SET.
THE CRITICAL PARAMETER IS "BEING MEANINGFUL".

c) DATA ACQUISITION : DATA ARE READ IN FROM THE APPARATUS, NORMALLY EQUIPPED WITH SPECIAL-PURPOSE INTERFACE MODULES AND STORED ON SOME PERMANENT MEDIUM FOR SUBSEQUENT ANALYSIS ; ADEQUATE BUFFERING MUST BE ORGANIZED TO KEEP UP WITH THE AVERAGE DATA PRODUCTION SPEED AND COPE WITH PEAK RATES. IF NECESSARY, MULTIPLE LEVELS OF SELECTION LOGIC (FILTERS) MUST BE AVAILABLE .
HERE , THE CRITICAL PARAMETER IS SPEED

⊢

REAL R.T.S. ALMOST INVARIABLY CONTAIN A MIXTURE OF THESE COMPONENTS .
THE STARTING POINT OF THE DESIGN LOGIC IS UNDERSTAND THEIR RELATIVE ROLE AND IMPORTANCE .

A NOTE ABOUT CONCURRENCY:

THE SIMPLEST TYPE OF S.P.S. IS THE ONE RUNS ON A SINGLE PROCESSOR.

THIS ACTS ON A SINGLE PROCESS AT ANY GIVEN TIME, I.E. IS TIME-MULTIPLEXED

THE PROCESSES ARE CALLED

QUASI-CONCURRENT (N. WIRTH)

A MORE COMPLEX KIND IS THE ONE BASED ON SEVERAL IDENTICAL PROCESSORS

THE PROCESSES ARE CALLED

CONCURRENT

A THIRD KIND IS THE ONE IN WHICH WE HAVE DIFFERENT KINDS OF PROCESSORS (I.E. SPECIALIZED PROCESSORS), E.G. I/O PR., COPROCESSORS (FL. PT.) ...

IT IS IMPORTANT TO WORK AS FAR AS POSSIBLE, AT THE HIGHEST LEVEL OF ABSTRACTION, TRYING INTO EVIDENCE THE CONCEPTS COMMON TO ALL KINDS.

SO, WE CAN DEFINE IN ANY OF THIS CASE, THE FOLLOWING CONCEPTS:

A PROCESS CAN BE { STARTED . SUSPENDED RESUMED

PROCESSES COMMUNICATE VIA { SHARED VARIABLES SIGNALS (MESSAGES)

SIGNALS (NORMALLY) CARRY NO DATA

A PROCESS (SUSPENDED) WAITS FOR A SIGNAL

A SIGNAL SHALL AWAKE AT MOST ONE PROCESS

A MONITOR IS A MODULE THAT HANDLES SHARED DATA SO AS TO INSURE THEIR INTEGRITY AND DISTRIBUTES SIGNALS SO AS TO INSURE "FAIRNESS" AND TO AVOID "STARVATION" AND "DEADLOCK"

VERY GENERAL STRUCTURE OF A R.T.S.

HERE ARE THE MAIN POINTS TO BE CAREFULLY ANALYSED BEFORE STARTING SYSTEM IMPLEMENTATION, AND, EVEN MORE CODE WRITING!

- HARDWARE SPECS : WRITE CLEAR DEFINITION OF THE HARDW. TO BE CONTROLLED, ESPECIALLY IN TERMS OF TIME DEMANDS : TRY TO FORM HOMOGENEOUS CLASSES OF MODULES, WITH SIMILAR CHARACT. THINK ABOUT HARDWARE TESTS BEFORE.

- CONTROL SOFTWARE : THE COST PARAMETERS OF THE SYSTEM AND THE COMPLEXITY OF THE HARDWARE WILL HELP TO DEFINE A RANGE OF CONTROL COMPUTERS :
  → OPER. SYST, LANGUAGES, DEBUGGING TOOLS AVAILABLE SHOULD BE CONSIDERED
  DO NOT UNDERESTIMATE THE REAL COST IN TIME TO DEVELOP AND EXISTING SUPPORT SOFTWARE.

- LINKS TO EXTERNAL WORLD : TWO REASONS :
  a) LARGE AMOUNTS OF DATA
  b) CROSS-SOFTWARE
  CAREFUL : YOU MAY NOT NEED IT TODAY, BUT A COMPLETELY CLOSED SYSTEM MAY BE A PROBLEM TOMORROW.
  REMEMBER:
  A GOOD QUALITY, STANDARD TAPE UNIT OR FLOPPY DISC + FAST HORSE IS A LINK TO THE EXTERNAL WORLD.

- USER INTERACTION - REMEMBER : SPEED, EFFICIENCY AND RELIABILITY ARE BASIC REQUESTS, BUT WHAT YOU ARE GOING TO LIVE WITH EVERY DAY IS THE USER INTERFACE.

- DEBUGGING - BE PESSIMISTIC : ANYTHING THAT CAN HAPPEN, WILL HAPPEN, AND MANY THINGS WHICH COULD NOT POSSIBLY HAPPEN, WILL HAPPEN. THINK IN ADVANCE ABOUT DEBUGGING TOOLS.

- UPGRADING AND MAINTENANCE
  - REAL-TIME-SYSTEMS ARE ALIVE, AND CONSTANTLY CHANGING. NEVER SAY: "... IF WE CHANGE THAT, I WILL HAVE TO REWRITE THE PROGRAM" : YOU WILL HAVE TO DO IT ANYHOW, AND YOU WILL LOOK INCOMPETENT.

EXAMPLES

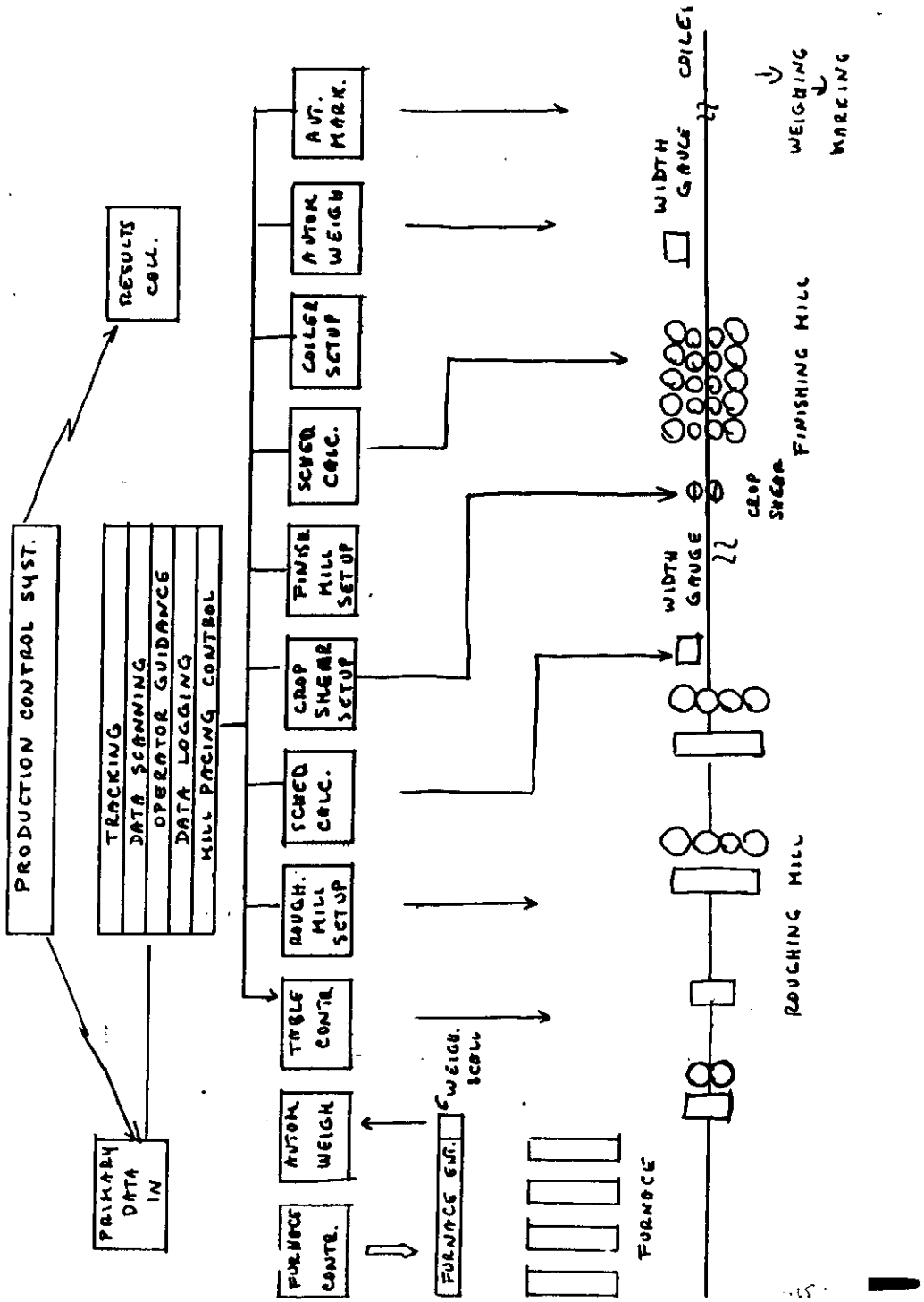
d) HOT STRIP MILL CONTROL – JAPAN.

SYSTEM DEMANDS: – PRODUCT QUALITY IMPROVEMENT
– ENERGY SAVING
– REAL-TIME HANDLING OF LARGE AMOUNTS OF DATA NECESSARY → (HEAT TRANSFER MODEL AND PLASTIC DEFORMATION MODEL).

WHEN ROLLING ORDERS ARE GIVEN, SLABS OF VARIOUS SIZES AND STEEL GRADES ARE CHARGED INTO THE REHEATING FURNACE AND HEATED UP TO THE REQUIRED TEMPERATURE. THEN THEY ARE ROLLED INTO THE ORDERED SIZES, FINISHED, COILED AND SENT TO A WEIGHING SCALE AND MARKING MACHINE

SYSTEM FEATURES:

- A VARIETY OF EQUIPMENT, SUCH AS REHEATING FURNACES, ROLLING MILLS, MEAS. INSTR., AUTOMATION EQUIPMENT IN A LONG LINE (900 m), MUST BE CONTROLLED
- LARGE AMOUNT OF DATA AT HIGH SPEED AND WITH HIGH RELIABILITY (MORE THAN 2000 PTS MEAS.)
- COMPLICATED CALCULATIONS IN REAL-TIME TO CONTROL THE FURNACE AND THE MILL.
- HIGH SPEED RESPONSE IN PROCESS CONTROL, OPERATOR INTERFACE AND DATA TRANSMISSION
- EQUIPMENT EXPANSION AND REMODELLING ARE CONTINUALLY PERFORMED, AND IT IS NECESSARY TO UPDATE THE COMPUTER SYSTEM CORRESPONDENTLY. THUS, SYSTEM MAINTENANCE SHOULD BE EASY, AND SYST. FLEXIBLE.
- IN CASE OF REPLACEMENT OF THE COMPUTER, PRODUCTION



SYSTEM IMPLEMENTATION:

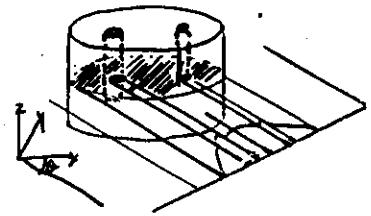
COMPUTER CONTROL RANGES FROM REHEATING FURNACE TO SLAB AND COIL TRACKING, OPERATOR INTERACTION, DATA LOGGING AND PROCESS CONTROL.

THE COMPUTER IS AN INTEGRATED SYSTEM OF DIFFERENT TYPES OF PROCESSORS (T-7/70, T-40C) AND MICROPROC.: THESE LAST ONE ARE DEVOTED TO THE PROCESS CONTROL (TEMPERATURE, GAUGES, CROP SHEAR ...) WHILE THE BIGGER ONES CONTROL THE MILL SETTINGS AND THE COMPUTATIONS FOR HEAT TRANSFER MODE.

TYPICAL SCANNING PERIODS FOR DATA RANGE FROM MAX OF 20 SEC TO MINIMUM OF 400 MSEC

HIGH-SPEED DATAWAYS 10 M BITS/SEC INSURE DISTRIBUTED CONTROL.

2) CAMAC CONTROL A'D ACQUISITION SYSTEM FOR X-RAY INDUSTRIAL TOMOGRAPHY.

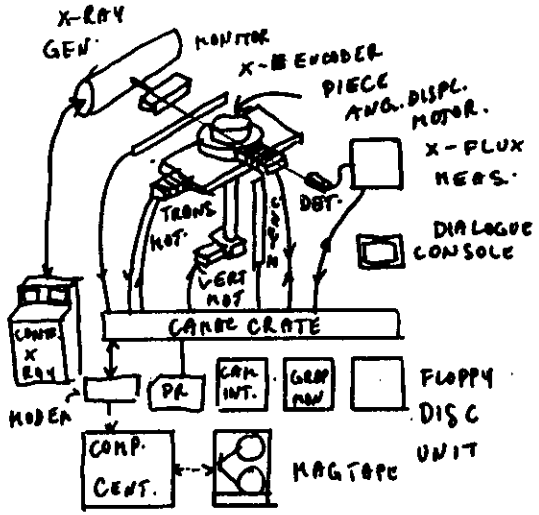


X-RAY BEAM (400KV 8V) IS SENT THROUGH THE OBJECT AND DETECTED. THE TRANSMITTED BEAM INTENSITY IS MODULATED BY THE INTEGRATED DENSITY ALONG THE CHOSEN DIRECTION

THE PROFILE IS DIGITIZED AND RECORDED, FOR VARYING θ VALUES AND SLICES.

THE PROBLEM OF SYSTEM CONTROL INCLUDES:

- CONTROL OF X-BEAM TUBE
- " " DETECTOR
- " MOTOR FOR ELEMENT DISPLAC. & ROTATION
- ACQUISITION OF DATA
- ON-LINE DISPLAY OF PROFILES
- OPERATOR INTERACTION FOR PIECE SET-UP
- STORAGE OF DATA ($\sim 10^6$ WORDS / PIECE)
- 3D RECONSTR. OF DENSITY (NOW OFF-LINE) ←



THE CONTROL SYSTEM INCLUDES 3 μPROC INTEL 8080 INTEGRATED INTO SPECIAL PURPOSE UNITS

THE FIRST TAKES OVER THE GENERAL SYST. ORG.

THE SECOND, FLOPPY DISK AND THE THIRD MAG. TAPE INTERF.

→ A SPECIAL-PURPOSE MODULE CONTROLS THE MOTORS, THE ENCODERS AND THE X-RAYS DETECTOR (SIGNAL IS TREATED WITH A AD/SHA 1144 SAMPLE AND HOLD ($1\ \mu sec\ 3 \cdot 10^{-5}$) FOLLOWED BY A 14 BIT ADC AD/ADC 1130 (25 μ)

PROGRAMMING LANGUAGE IS REAL-TIME BASIC

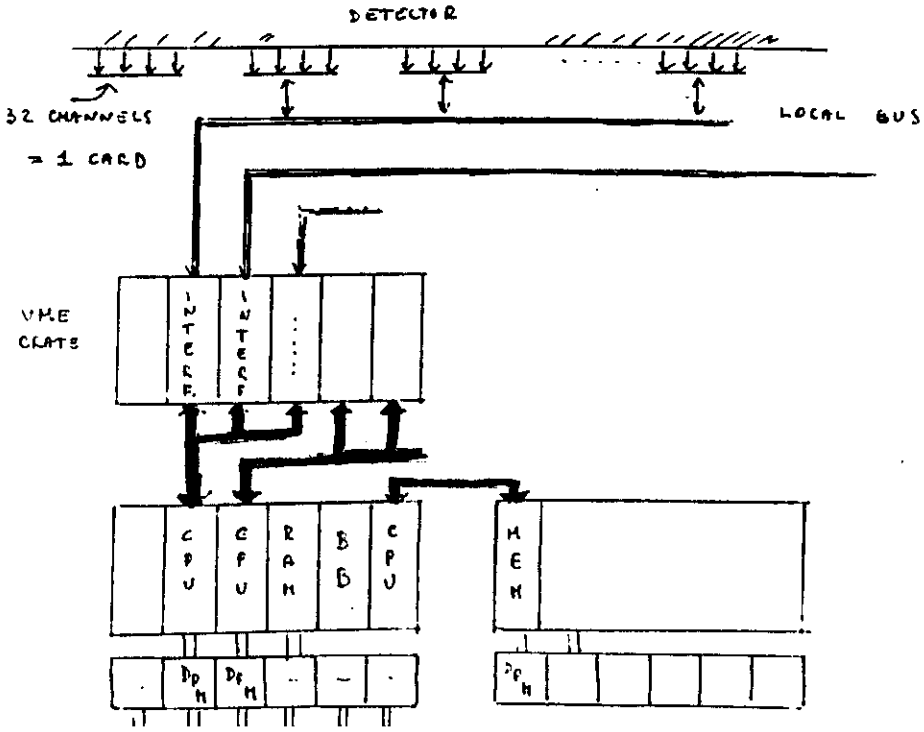
A FEW PARTS ARE WRITTEN IN ASSEMBLER.

3) CONTROL AND DATA ACQUISITION FOR A H.E.P. DETECTOR.

- CERN

SYSTEM DEMANDS:

- CONTROL OF PURPOSE BUILT HIGH SPEED ELECTRONICS
- VERY FAST DATA ACQUISITION ≈ 50,000 CHANNELS IN 47 MS.
- LARGE DATA REDUCTION
- INTERACTION WITH COMPLEX D.A.S.
- MONITORING AND SAMPLING OF DATA ON-LINE.



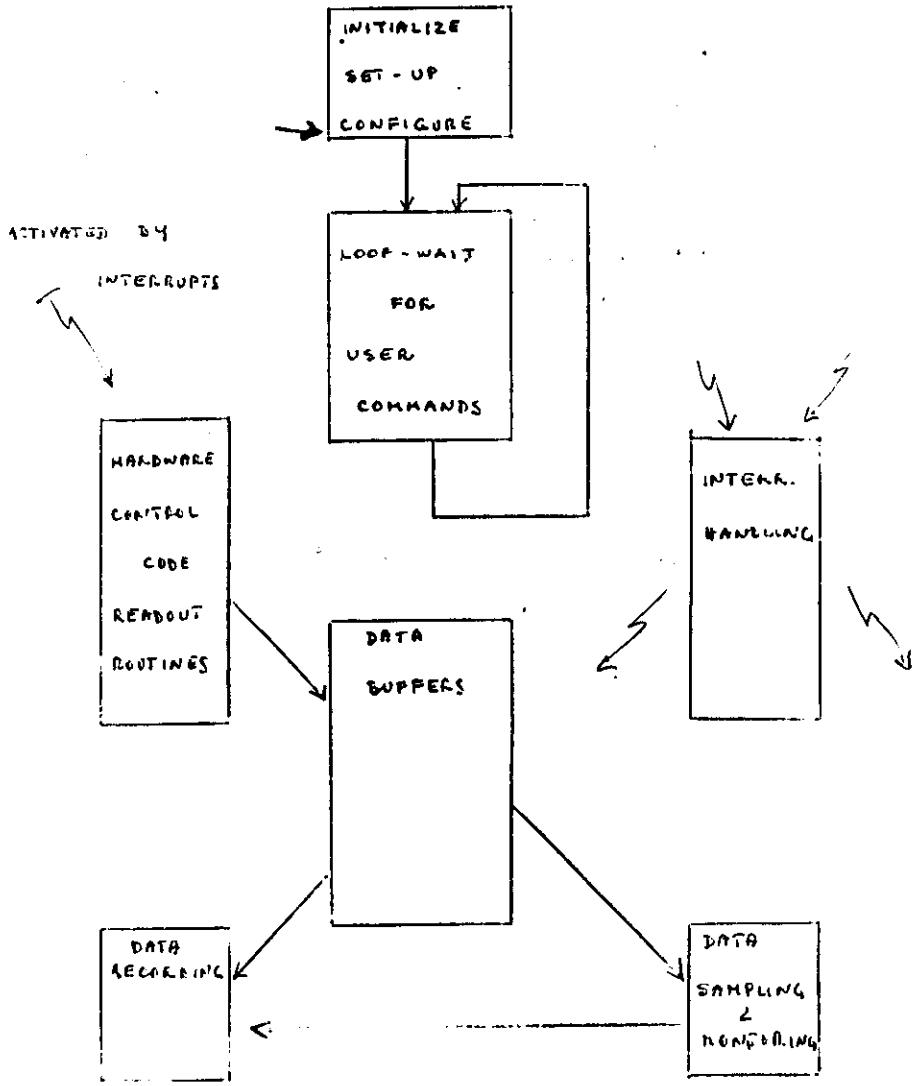
SYSTEM CONTROL AND DATA ACQUISITION ARE REALIZED IN THE FRAMEWORK OF VME STANDARD: PURPOSE - BUILT INTERFACE MODULES CONNECT THE LOCAL BUS TO THE CPUS (M. 68010) THROUGH THE AUXILIARY VMX BUS.

CPUS COMMUNICATE BETWEEN THEM AND WITH DIFFERENT KINDS OF MEMORY BOARDS THROUGH DPMS VISIBLE TO THE VME BUS.

"EVENTS" ARE WRITTEN ON A FIFO AND ASYNCHRONOUSLY SAMPLED AND ANALYSED BY A MONITORING CPU.

SYSTEM FEATURES INCLUDE AUTOM. BOOTSTRAPPING AND RESTART ON FAILURES (BUS ERROR AND SIMILAR), EASY RE-CONFIGURATION THROUGH SOFTWARE TABLES, SUB-SYSTEM INDEPENDENCE FROM MAIN D.A.Q.

ORGANIZATION OF A R.T.S. ON A TRADITIONAL COMPUTER



ORGANIZATION OF A R.T.S. ON A TRADITIONAL COMPUTING SYSTEM

IN THE '60S AND '70, THE MINICOMPUTER, BORN AS A LABORATORY COMPUTER, HAS EVOLVED TO BECOME A QUITE POWERFUL AND SOPHISTICATED INSTRUMENT FOR R.T.S. CONTROL.

THE TYPICAL STRUCTURE OF A R.T.S. INCLUDES:

A. HARDWARE CONTROL AND DATA ACQUISITION

- WHERE HARDWARE INTERFACE STANDARDS WHERE ADOPTED, POWERFUL LIBRARIES OF SUBROUTINES OR MACROS.

EX: FORTRAN CALL:

→ CALL CDREG (, ,)
CALL CSSA (, ,)

ASSEMBLER

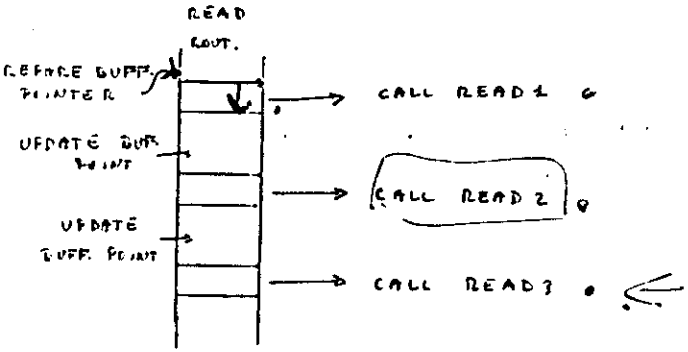
OR INTERMEDIATE LANGUAGE (ES. NPL)

* RS CRATE, STATION, SUBADDR....
WC; * DMA CRATE1, CRATE2, STATION1, STATION2
.....
END;

ON THE BASIS OF THESE LIBRARY CALLS, IT IS EASY TO BUILD UP THE PART OF THE PROGRAM THAT DEALS WITH DATA ACQUISITION OR

THERE ARE ESSENTIALLY TWO TECHNIQUES OF ORGANIZING THE READOUT AND HARDWARE CONTROL SOFTWARE.

1) STANDARD SUBROUTINE OR TASK ORGANIZATION.



THE STEERING TASK OR SUBPROGRAM ACTIVATES IN TURN MODULI RESPONSIBLE FOR READOUT OF DIFF. PARTS OF THE SYSTEM, PROVIDING TO EACH A POINTER TO THE POSITION IN THE BUFFER WHERE DATA CAN BE PUT, AND A MAX. WC.

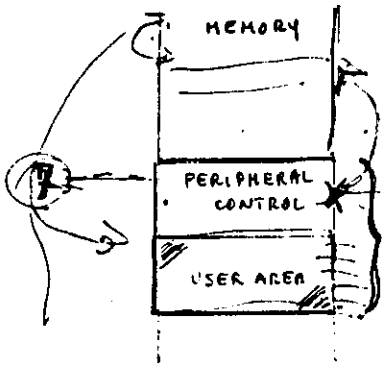
2) "LIST" ORGANIZATION.

THIS TECNIQUE IS BETTER WHEN THE HARDWARE SET UP IS FREQUENTLY CHANGING BUT THE TYPE OF MODUL IS STANDARD AND THE OPERATIONS NECESSARY FOR READOUT AND CHECKS ARE SIMPLE AND WELL DEFINED.

IN THIS CASE WE HAVE A SINGLE ACQUISITION TASK AND THE APPARATUS IS DESCRIBED IN A LIST WHICH IS USED BY THE TASK AS STEERING DATA.

CHANGING THE HARDWARE DOES NOT REQUIRE MODIFICATIONS IN THE PROGRAM.

AN ELEGANT SOLUTION TO THE PROBLEM OF INTERACTING WITH THE HARDWARE IS PROVIDED BY SYSTEMS HAVING MAPPED I/O:



"EXTERNAL PAGE"

IN THIS SYSTEMS, I/O OPERATIONS DO NOT REQUIRE SPECIAL INSTRUCTIONS. WHAT DEFINES THEM AS I/O RATHER THAN NORMAL MEMORY REFERENCE OPERATIONS IS THE ADDRESS.

A "WINDOW" OR "EXTERNAL PAGE" IS MAPPED ON PERIPHERAL DEVICES' CONTROL & STATUS REGISTERS OR DATA REGISTERS.

IN ASSEMBLY LANGUAGE, IT IS ENOUGH TO REFER TO ABSOLUTE ADDRESSES TO GAIN ACCESS TO PERIPHERALS OR TO USER HARDWARE.

IN HIGH LEVEL LANGUAGES STANDARD TECNIQUES EXIST TO EQUIVALENCE F.I. A VECTOR TO ONE SUCH "EXTERNAL AREA".

THE CLASSICAL PROBLEM OF HARDWARE CONTROL AND DATA ACQUISITION IS THE FACT THAT ONE WORKS INTO AN OPERATING SYSTEM ENVIRONMENT AND YET, FOR FAST RESPONSE ONE OFTEN WANTS TO BYPASS THE STANDARD PROCEDURES OF TASK ACTIVATION, QUEING, ETC, I.D. "THE PRICE ONE HAS TO PAY TO BE PROTECTED".

SOLUTIONS :
- WORK IN "KERNEL MODE"
- WORK IN "DIRECT MODE"
- HIGH PRIORITY INTERRUPT

ALL THESE TECNIQUES ARE DANGEROUS SINCE THIS STATES THE USER PROGRAM IS NOT PROTECTED. AN ERROR THAT WOULD NORMALLY RESULT IN A M.O.S. MESSAGE WILL (OR MAY) CAUSE A CATASTROPHIC FAILURE.

SO : THIS PART OF THE CODE SHOULD BE KEPT OF MINIMAL LENGTH, AND WRITTEN WITH GREAT CARE.

IT IS ALMOST INVARIABLY WRITTEN EITHER IN ASSEMBLY LANGUAGE OR IN ONE OF THE EQUIVALENT "INTERMEDIATE LANGUAGES" WHICH DO NOT HIDE THE COMPUTER ARCHITECTURE.

INTERRUPTS : THE ABILITY TO EFFICIENTLY HANDLE INTERRUPTS, THE INT. RESPONSE TIME, THE NUMBER OF PRIORITY LEVELS AND THE POSSIBILITY OF DIRECT TRAPPING TO INTERRUPT SERVICE ROUTINES ARE THE BASIC PARAMETERS TO JUDGE THE R.T.S. POWER

- INTERRUPT HANDLING INVOLVES CONTEXT SWITCHING, WHICH IS NORMALLY DONE HARDWARE.
- THE STANDARD USE OF I. IS TO PROVIDE FAST RESPONSE TO ASYNCHRONOUS EVENTS (DATA ARRIVAL, ALARMS, ERRORS, POWER FAILURE ...) WHILE GIVING THE SYSTEM THE POSSIBILITY TO PERFORM BACKGROUND ACTIVITY WHILE WAITING FOR THESE EVENTS
- INTERRUPTS MUST BE CONNECTED, I.E. THE SYSTEM MUST BE INFORMED OF THE NAME OF THE TASK OR SUBROUTINE THAT WILL HANDLE THE INT., OF THE SOFTWARE PRIORITY OF IT AND OF OTHER CONTEXT PARAMETERS
- INTERRUPTS CAN BE MASKED; NORMALLY THE INT. OF THE SAME OR LOWER PRIORITY LEVEL AS THE ONE BEING NOW HANDLED ARE HARDWARE MASKED OFF.

INTERRUPT HANDLING IS THE MOST DELICATE PART [illegible]

SINCE IT CAN GENERATE THE WORST KIND OF BUG:

→ THE TIME-DEPENDENT BUG.

A COMPLICATE SCHEME OF SEVERAL PRIORITY LEVELS, HARDWARE PRIORITY VS SOFTWARE PRIORITY, WHEN TO MASK OUT AN INTERRUPT AND WHEN TO ENABLE IT, ISR THAT CHANGE THE SITUATION SO THAT THE INTERRUPTED PROGRAM DOES NOT FIND ITS WAY OUT, ETC...

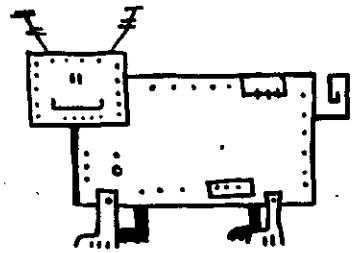
ABOVE ALL, TIME ENTERS HEAVILY INTO THE PICTURE. WE HAVE SEVERAL TASKS THAT CAN BE ACTIVATED OR DEACTIVATED AT RANDOM POINTS IN TIME, SO THAT THE EXACT SEQUENCE IN WHICH THEY WILL BE EXECUTED IS NOT KNOWN A PRIORI.

AS A RULE, THE ACTIONS OF THE ISR SHOULD BE KEPT SHORT, AND REDUCED TO:

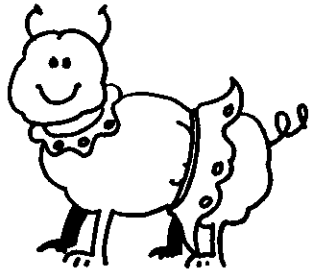
- DETERMINING THE CAUSE OF @ INTERRUPT
- PERFORMING URGENT ACTIONS (E.G. BUFFER SWITCH)
- SETTING FLAGS TO WARN THE OTHERS OF THE INTERRUPTION.
- RE-ENABLE INTERRUPTS AND RETURN AS SOON AS POSSIBLE

2

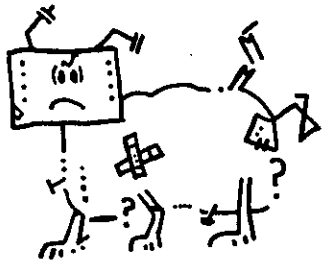
AND, SINCE WE STARTED TALKING ABOUT BUGS, LET US BEGIN BY CLASSIFYING THEM



TYPICAL "HARDWARE" BUG



TYPICAL "SOFTWARE" BUG



TYPICAL NASTY BUG

A TYPICAL "HARDWARE" BUG CAN BE DIFFICULT TO FIND OR CURE, BUT IS CLEARLY CONNECTED TO THE APPARATUS OR TO THE INTERFACE ELECTRONICS. IT APPEARS SIMILAR TO BOTH PRODUCTION SOFTWARE AND TO HARDWARE TEST SOFTWARE. IT DISAPPEARS BY CHANGING MODULUS OR POSITION IN THE CHASSIS ETC...

A DIFFICULT KIND IS CREATED BY "THE DYING COMPONENT EFFECT", WHERE ONE CAN HAVE APPARENT TIME-DEPENDENT BEHAVIOURS THAT ARE REALLY TEMPERATURE-DEPENDENT ONES.

A TYPICAL "SOFTWARE" BUG NORMALLY DEVELOPS JUST AFTER AN "IMPROVEMENT" IN THE CODE THAT IS NORMALLY "MINOR" AND IN A REGION "UNCORRELATED" TO THE APPARENT SOURCE OF ERROR.

A STANDARD TECNIQUE IS TO GO BACK TO THE OLD PROGRAM VERSION.

IN SPECIAL CASES (E.G. STACK HANDLING, SEE LATER) SOFTWARE BUGS TEND TO FALL INTO THE THIRD CATEGORY, I.E.

NASTY BUGS : THESE TYPICALLY LOOK SOFTWARE WISE TO HARDWARE EXPERTS AND VICEVERSA. THE VICIOUS ONES HAVE NORMALLY TO DO WITH TIMING AND THEREFORE HAVE THE TENDENCY TO DISAPPEAR OR CHANGE WHEN EXAMINED BY THE STANDARD TECNIQUES: SINGLE STEP, DEBUG PRINTING, SYMBOLIC DEBUGGING TECNIQUES.

THE ONLY WEAPON I KNOW AGAINS THIS TYPE OF BUG IS USING DYNAMIC TECNIQUES, SUCH AS TRACING.

PROVISION FOR DEBUGGING SHOULD BE MADE ALREADY AT THE SYSTEM DESIGN LEVEL:

- FOR NON-TIME-DEPENDENT PROBLEMS, SEVERAL LEVELS OF DEBUG PRINTOUTS, ACTIVATED BY FLAGS, USED WITH THE "CONDITIONAL COMPILING" AND "CONDITIONAL ASSEMBLY" FEATURES.
- FOR TIME-DEPENDENT PROBLEMS, VERY FAST AND EFFICIENT CHECKS AND TRACING THAT CAN REMAIN IN THE CODE ALL THE TIME.

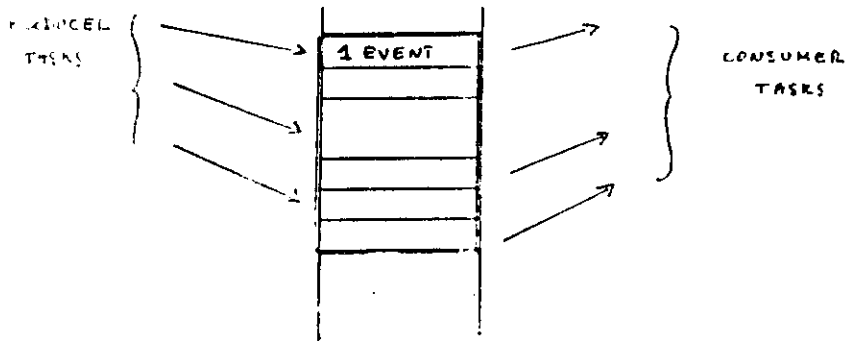
DATA BUFFERS:

WHEN THE R.T.S. IS HEAVILY ORIENTED TOWARDS DATA ACQUISITION, THE PROBLEM OF BUFFERS' HANDLING BECOMES IMPORTANT.

IF THE RATE AT WHICH DATA IS PRODUCED, I.E. GENERATED, AND THAT AT WHICH IT IS CONSUMED, I.E. ANALYSED AND/OR RECORDED, DO NOT MATCH, WE MUST STORE DATA IN MEMORY OR INTO HARDWARE BUFFERS.

IF PEAK RATES OF PRODUCTION ARE LARGER THAN CONSUMING RATES, BUT ON AVERAGE WE ARE FAST ENOUGH, BUFFERING IS SUFFICIENT TO DERANDOMIZE DATA PRODUCTION.

IF, ON THE OTHER HAND, THE AVERAGE RATE IS TOO HIGH, WE MUST FILTER DATA, I.E. SELECT VERY QUICKLY A FRACTION OF IT THAT WE CAN STORE. ON A UNIPROCESSOR MACHINE THIS IS VERY DIFFICULT BECAUSE THE FILTERING TASK MUST COMPETE WITH THE INPUT TASK FOR CPU TIME



DETECTING FOR OPTIMAL SIZE OF THE DATA BUFFER

CONFIGURING THE SYSTEM MEANS DEFINING TO

THE SUPERVISOR TASK :

- THE NUMBER AND SIZE OF DATA BUFFERS
- THE IDENTITY OF THE PRODUCER TASKS, I.E. TASKS WHICH HAVE "WRITE ACCESS" TO THE BUFFER(S)
- THE IDENTITY OF CONSUMER TASKS, I.E. TASKS WHICH HAVE "READ ACCESS" TO THE BUFFERS. FOR EACH OF THESE, A "SAMPLING FRACTION" CAN BE DEFINED.

DATA BLOCKS TO BE "CONSUMED" ARE MARKED AND CANNOT BE DELETED UNTIL USED.

A CONSUMER TASK WITH A FIXED SAMPLING FRACTION AND WHICH DIES OR REMAINS IN A LOOP IS THE SIMPLEST AND FASTEST WAY TO BLOCK DATA ACQUISITION.

THE PRODUCER-CONSUMER RELATIONSHIP IS THE MOST DIRECT EXAMPLE OF "SHARED RESOURCE" PROBLEM. FOUR MAJOR PROBLEMS MUST BE ANALYSED IN THIS RESPECT :

- MUTUAL EXCLUSION: IS STRICT SEQUENTIAL USE OF A RESOURCE BY COMPETING PROCESSES.

- SYNCHRONIZATION: COOPERATING PROCESSES REQUIRE SYNCHR. IN ORDER TO COORDINATE A SEQUENCE OF OPERATIONS ON SHARED RESOURCES

- DEADLOCK : (ALSO CALLED "DEADLY EMBRACE") - THIS TERM DESCRIBES A SITUATION IN WHICH TWO PROCESSES ARE MUTUALLY WAITING.

- DETERMINACY : THIS MEANS THE POSSIBILITY THAT PROCESSES WHICH CAN CONCURRENTLY ACCESS COMMON DATA COULD ALTER THIS DATA BY AN INTERLEAVED SEQUENCE OF OPERATIONS WHICH INVALIDATES IT. SPECIAL PRECAUTIONS ARE NECESSARY TO ENSURE THAT COMMON DATA IS DETERMINATE.

THE KEY TO ORDERLY INTERACTION OF CONCURRENT PROCESSES, NOT ONLY IN THE PRODUCER/CONSUMER PROBLEM BUT IN THE GENERAL CASE, IS THE CONCEPT OF CRITICAL REGION :

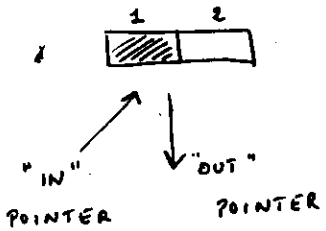
IT IS A SECTION OF CODE IN A CONCURRENT SYSTEM WHICH IS CONTROLLED SO THAT ONLY ONE PROCESS AT A TIME CAN ENTER.

CONCURRENT SYSTEMS USE THE SEMAPHORE CONCEPT TO CONTROL ACCESS TO CRITICAL REGIONS.

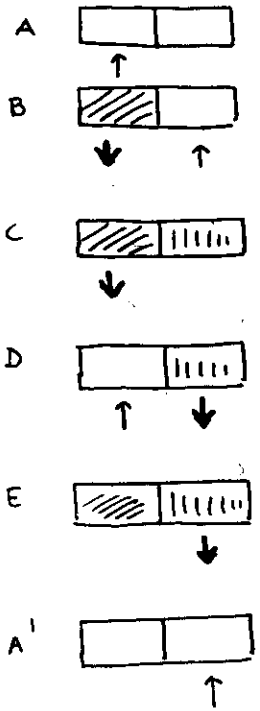
- INDIVISIBLE OPERATIONS
- BUSY WAIT
- SEMAPHORE

EXAMPLE -

THE DOUBLE BUFFER PROBLEM.



POSSIBLE STATES OF THE D.B. SYSTEM:



STATE DIAGRAM TRANSITIONS

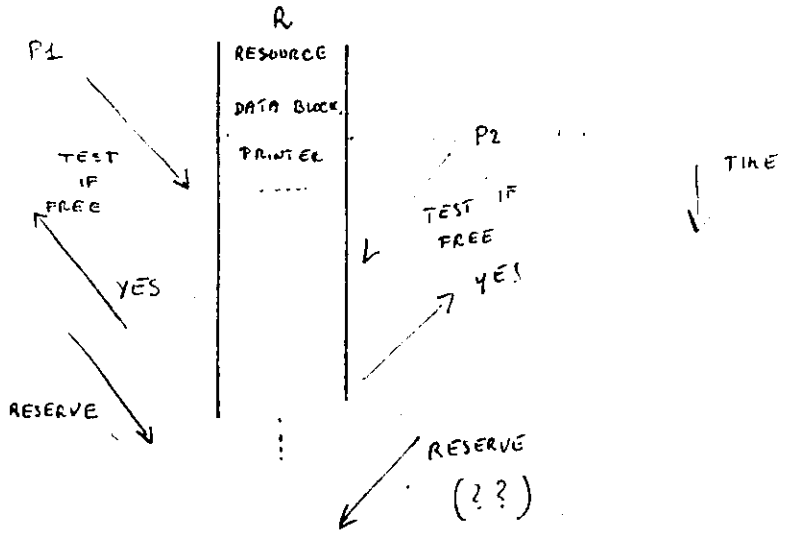
SUPPOSE YOU ARE IN STATE B AND READING OUT DATA FROM BUFFER 1. A NEW EVENT COMES IN AND IT IS STORED IN BUFFER 2. IF YOU WAIT FOR END OF READ TO ACKNOWLEDGE THIS ARRIVAL, THE TRANS. WILL BE:

B → A → B WRONG!

INDIVISIBLE OPERATIONS

A SEQUENCE OF CODE WHICH CANNOT BE INTERRUPTED EITHER SOFTWARE OR HARDWARE.

CONSIDER THE FOLLOWING EXAMPLE:



PROCESS P1 TESTS A FLAG TO DETERMINE WHETHER R IS AVAILABLE; SUPPOSE IT IS SO; IN A SUCCESSIVE OPERATION P1 SETS THE FLAG TO INDICATE THAT R IS NOW NOT FREE. IF ANOTHER REQUEST ARRIVES BETWEEN THESE TWO EVENTS, THE RESULT IS AN ERROR.

SUPPOSE THE FLAG IS EXAMINED AND MANIPULATED BY AN INDIVISIBLE PROCEDURE, AS FOLLOWS:

CP: [IF FLAG = FALSE THEN FLAG := TRUE]

BUSY WAIT IF NOT

AND RESET BY ANOTHER INDIVISIBLE OPERATION

[FLAG := FALSE]

THIS COULD SOLVE OUR PROBLEM.

THE PROBLEM WITH THIS SOLUTION IS THAT THE CPU IS OCCUPIED IN TESTING THE FLAG AND THIS IS A NON-PRODUCTIVE ACTIVITY.

BETTER TECNIQUES IMPLY THAT THE PROCESS IS SUSPENDED WHILE WAITING FOR ITS RESOURCE, AND ANOTHER PROCESS CAN USE THE CPU TIME.

ON "TRADITIONAL" COMPUTERS, THE OPERATING SYSTEM OFFERED A VARIETY OF FACILITIES TO ALLOW THIS KIND OF GAME. EX

```
WAITEF (32)          ; WAIT FOR EVENT FLAG
                       # 32 TO BE SET
                       (TASK IS SUSPENDED)
SETEF (32)           ; SET EVENT FLAG 32
Q.IO (LUN, , BUFFER, ..., 32, ...) ; INITIATE I/O ACTIVITY
                       ON UNIT LUN; SET
                       EVENT FLAG 32 WHEN
                       DONE
TSTEF (32)           ; TEST EV. FLAG 32
```

THE SEMAPHORE IS A GENERALIZATION OF THE CONCEPT OF FLAG FROM THE PREVIOUS EXAMPLE.

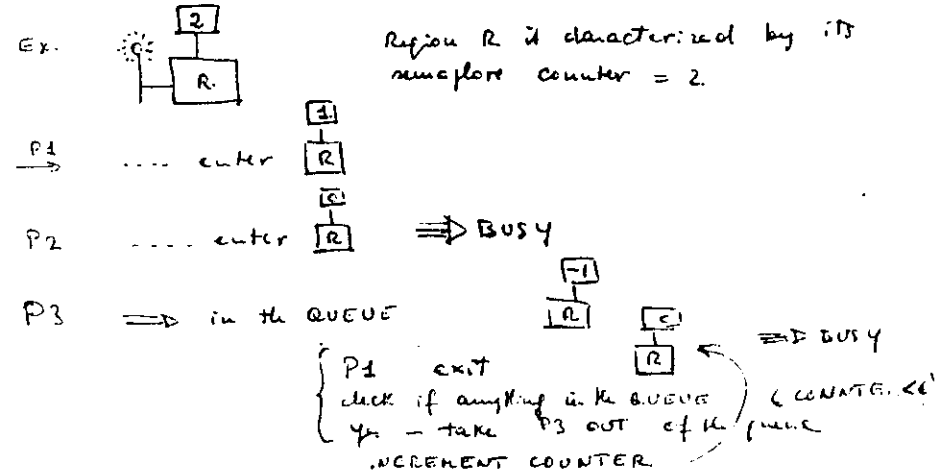
IT CONTAINS A QUEUE AND A COUNTER

THE COUNTER INDICATES THE NUMBER OF PROCESSES THAT CAN PASS THE SEMAPHORE WITHOUT BEING SUSPENDED OR HAVING TO WAIT.

FOR CRITICAL REGIONS, THE SEMAPHORE COUNT IS 1. IT GOES TO 0 WHEN THE CRITICAL REGION IS BUSY.

IF A PROCESS DEMANDS PASSAGE THROUGH THE SEMAPHORE AND THE COUNTER IS 0, IS IS PUT IN THE QUEUE OF SUSPENDED PROCESSES FOR THIS REGION.

BY ALLOWING THE COUNTER TO TAKE NEGATIVE VALUES, WE CAN RECORD ALSO THE NUMBER OF SUSPENDED PROCESSES.



STACKS

AS IT IS WELL KNOWN, A STACK IS A DEVICE THAT ALLOWS STORAGE OF OBJECTS WITH RETRIEVAL HAPPENING IN REVERSE ORDER (LIFO).

AS IT IS ALSO WELL KNOWN TWO BASIC OPERATIONS ARE DEFINED FOR STACK MANIPULATION:

PUSH : PUT AN ITEM ON THE STACK
POP : GET AN ITEM OUT OF THE STACK.

STACKS MAY BE BUILT WHICH GROW TOWARDS SMALLER ADDRESSES OR TOWARDS LARGER ADDRESSES.

NORMALLY STACK OPERATIONS INVOLVE THE USE OF

AUTOINCREMENT / AUTODECREMENT

ADDRESSING MODES, SO THAT THE "STACK POINTER" (NORMALLY A SPECIAL REGISTER) IS, AFTER AN OPERATION, AUTOMATICALLY READY FOR THE NEXT.

R.T.S. MAKE LARGE USE OF THE STACK, MAINLY FOR

- STORAGE OF TEMPORARY DATA ⟶ MEMORY SP. SAVING
- SUBROUTINE CALLS ⟶ PARAMETER PASSING & RET. ADDR.
- INTERRUPT HANDLING ⟶ CONTEXT SWITCHING
- LOCAL VARIABLES STORAGE ⟶ REENTRANT CODE.

P.I.C.

THE USE OF REENTRANT CODE IN PARTICULAR IS IMPORTANT, ESPECIALLY FOR TASKS FULFILLING THE ROLE OF "OPERATING SYSTEM", I.E. PROVIDING "SERVICE ROUTINES" WHOSE USE CAN BE REQUESTED BY SEVERAL USERS IN A NON-CONSECUTIVE WAYS.

REMEMBER : STACK HANDLING IS ANOTHER CRITICAL POINT IN R.T.S : ERRORS CAN LEAD TO VERY DIFFICULT

AT CRITICAL POINTS IN THE PROGRAM OR ISSUING DIAGNOSTIC MESSAGES IF CERTAIN CHECKS ON STACK POINTER VALUES ARE NOT FULFILLED.

D. DATA-BASES FOR MONITORING.

ALTHOUGH DATA BASE MANAGEMENT DOES NOT NORMALLY PLAY AN IMPORTANT ROLE IN R.T.S. THERE ARE INSTANCES WHERE THIS CAPABILITIES, RARELY IN EXTREME CONDITIONS, ARE NECESSARY:

- SUPPOSE WE HAVE A LARGE APPARATUS, COMPOSED OF SEVERAL FUNCTIONALLY DIFFERENT UNITS.

  FOR EACH UNIT, A SET OF PARAMETERS DETERMINES OPTIMAL WORKING POINTS, AND/OR CRITICAL OR ALARM CONDITIONS.

  A DATA BASE WOULD DESCRIBE THE SITUATION,

  ALLOWING NOT ONLY STORAGE OF THESE PARAMETER

  VALUES, BUT ALSO MEANINGFUL GROUPINGS AND

  RELATIONS BETWEEN THEM.

  (E.G. HIGH VOLTAGE CONTROL OF ≠ UNITS

  WORKING SET POINTS FOR TESTS - ETC..)

IN THESE CASES, IT IS PART OF THE R.T.S. CODE TO:

- SET-UP AND UPDATE THE D.B.S. CONTENT
- ROUTINELY SCAN THE HARDWARE UNITS COMPARING THE REAL VALUES WITH THE DESIRED ONES AND ISSUING DIAGNOSTIC, ALARMS & EMERGENCIES INTERLOCKS
- ALLOW USER INTERACTION FOR TEST & LOGGING INFO. INSPECTION

E. HOW TO MAKE THE R.T.S. USER FRIENDLY.

ONE POINT THAT THE ENTHUSIASTIC DESIGNER OF A BRAND NEW R.T.S. TENDS TO OVERLOOK IS THAT, ONCE THE SYSTEM IS WRITTEN, TESTED AND DEBUGGED, AND IN ROUTINE USAGE, HE WILL PROBABLY BE IN A VERY REMOTE PLACE, ENJOING THE HARD-EARNED MONEY, SO, THE AVERAGE USER OF THE SYSTEM WILL OFTEN NOT KNOW IN GREAT DETAIL HOW THE SYSTEM IS MADE.

LEAVING ASIDE DOCUMENTATION, THAT IS OFTEN LOOKED UPON WITH SUSPICION IF TOO WELL DONE AND TOO COMPREHENIVE, THE BEST IS TO GIVE THE SYSTEM SOME INTRINSIC CAPACITY OF MAKING ITSELF UNDERSTOOD.

- MENUS ARE A STANDARD WAY OF INTERACTING (OFFER THE POSSIBILITY OF NOT USING THEM TO EXPERIENCED USERS)
- DIAGNOSTICS SHOULD BE MEANINFUL AND EXPLICIT (OFFER THE POSSIBILITY OF CONCISE MESSAGES TO EXPERIENCED USERS)
- INFORM THE USER IF THE SYSTEM IS BUSY, SO HE WILL NOT THINK HE IS LOST IN A LOOP....
- USE REASONABLE DEFAULT, MINIMIZING CONSOLE TYPING.
- AVOID CATASTROPHIC ACTIONS AS A CONSEQUENCE OF MISTYPING. (THE "ARE YOU SURE ?" APPROACH)
- AVOID SUGGESTING POSSIBLE CAUSES OF FAILURE. (VERY IRRITATING IF WRONG, MILDLY IRRITATING IF RIGHT ....)

HOW TO MAKE THE R.T.S. GROW OLD GRACEFULLY

AS TECHNOLOGY PROGRESSES, THE R.T.S. WILL BECOME OLD AND OBSOLETE. THE "NATURAL" WAY OF AGEING IS "BECOMING INADEQUATE TO THE DEMANDS OR LESS EFFICIENT THAN NEW AND CHEAPER SOLUTIONS NOW AVAILABLE."

BADLY DESIGNED OR INSUFFICIENTLY THOUGHT OF SYSTEMS TEND TO AGE TOO SOON: IF NO DEBUGGING PROVISIONS WERE MADE, THEY ARE FILLED WITH "AD HOC" PATCHES, WITH FLAGS PLUGGED IN LOGICALLY UNCORRELATED BLOCKS WHERE "A FREE LOCATION" COULD BE FOUND," BY HORRIFYING "IF" STATEMENTS TAKING CARE OF VERY SPECIAL ERROR CONDITIONS WHICH HAPPENED ONCE AND NEVER AGAIN.

THE LOGICAL FLOW OF THE MAIN STEERING TASK TENDS TO BECOME BLURRED, AND IN SOME EXTREME CASES, FORGOTTEN CORPSES OF NUMERICAL CONSTANTS CAN BE FOUND IN THE MIDDLE OF THE CODE.

AVOIDING THIS DECAY IS ONE OF THE TASKS OF A THE SYSTEM DESIGNER. HINTS:

MODULARITY, SIMPLICITY OF THE LOGICAL FLOW, TOP-DOWN APPROACH, CLEAR DEFINITION OF INPUT, OUTPUT AND CROSS REFERENCE OF CODE BLOCKS, INTERNAL CODE DOCUMENTATION, PROVISION FOR DEBUGGING, DEFINITION OF CRITICAL POINTS WHERE TRACING CAN BE DONE, AVOIDING

KANY

MULTIPROCESSOR-BASED R.T.S.

IN HIGH-PERFORMANCE APPLICATION OF COMPUTERS FOR REAL-TIME CONTROL, THE SOLUTION ADOPTED NOW IS THE USE OF MULTIPLE PROCESSORS, SHARING RESOURCES FOR LOGICAL OR ECONOMIC NECESSITY.

THE LOW COST OF μPROC. ENCOURAGES THE DEVELOPMENT OF NEW CONFIGURATIONS WHICH WOULD BE UNECONOMICAL WITH CONVENTIONAL PROCESSORS.

THE CONCURRENCY PROBLEMS RESULTING FROM SUCH ARCHITECTURES ARE MORE COMPLEX THAN CONCURRENCY IN A UNIPROCESSOR SYSTEM.

AS FAR AS DESIGN GOES, ONE MORE LEVEL OF ANALYSIS IS NECESSARY, TO SELECT THE PROBLEM'S PARTITION AMONG THE VARIOUS PROCESSORS.

SO, AFTER HAVING ANALYSED THE PROBLEM IN ITS GENERALITY, WE MUST TRY TO DESCRIBE IT IN A WAY THAT EVIDENCES ITS LOGICAL CONCURRENCY.

THIS PARTITION MUST THEN BE PUT INTO RELATION WITH REAL HARDWARE ARCHITECTURE.

FINALLY, ONE SHOULD TRY TO DO PERFORMANCE EVALUATION TO SELECT OPTIMAL SOLUTIONS.

SELECTION OF THE BASIC FUNCTIONS (TASKS) AND PARTITIONING OF THE DATA, THESE CAN BE DIVIDED INTO THREE CLASSES:

a) "PRIVATE" DATA, INTERNAL TO A BLOCK (PROCESS)

b) "COMMON" BUT "STATIC" DATA - THEY CAN BE DUPLICATED

c) "COMMON" AND "ALTERABLE" DATA - TO BE ACCESSED WITH CARE.

THE ADVANTAGES OF SEPARATING THE SYSTEM INTO PARALLEL ACTIVITIES ON DIFFERENT PROCESSORS MUST BE WEIGHED AGAINST THE PRICE OF ESTABLISHING AMONG THESE PROCESSORS A FLOW OF DATA, COMMANDS AND MESSAGES AND OF SYNCHRONIZING THE PROCESSES ACTIVITY.

HARDWARE / SOFTWARE SPLIT

IF THE DESIGN IS COMPLETELY OPEN, ONE MIGHT HAVE TO DECIDE TASK SHARING BETWEEN HARDWARE AND SOFTWARE.

DESIGNING AND DEVELOPING SPECIAL PURPOSE HARDWARE SHOULD BE DONE WHEN EXISTING SOFTWARE OR HYBRID (PLU, ECL-ALU, TABLE LOOK-UP) TECHNIQUES ARE REALLY INADEQUATE OR WHEN A LARGE NUMBER OF MODULES WILL BE BUILT AFTER PROTOTYPE DEVELOPM.

* * * * * ..... *

AS A RULE, ONE SHOULD TRY TO BUILD THE SYSTEM OUT OF A SMALL NUMBER OF DIFFERENT COMPONENTS: COMMERCIALLY AVAILABLE ONES CAN BE REPLACED AS

MASTER / SLAVE OR DEMOCRACY ?

AS WE HAVE PREVIOUSLY DISCUSSED, A R.T.S. IS ALWAYS A MULTI-PROCESS ONE: IT MUST INCLUDE TECNIQUES TO ALLOW A PROCESS TO BE SUSPENDED OR RESTARTED, TO WAIT FOR AN EVENT, TO START ANOTHER PROCESS ETC...

A TYPICAL PARTITION ON A UNIPROCESSOR MACHINE WOULD BE:

- ONE OR MORE (USUALLY ONE) DATA ACQ. PROC.
- ONE OR MORE BACKGROUND "CONSUMER" PROCESSES
- A SUPERVISOR PR. TO DEAL WITH RESOURCE SHARING
- ONE OR MORE HIGH PRIORITY PROCESSES TO BE ACTIVATED ON INTERRUPT.

HAVING AT OUR DISPOSAL SEVERAL INDEPENDENT PROCESSORS WE WOULD NATURALLY SPLIT THEM INTO PRODUCERS AND CONSUMERS, AND ALSO SEPARATE HIGH PRIORITY PROCESSES FOR CONTROL, ALARMS ETC...

IN AN IDEAL CASE, THESE PROC. COULD WORK TOGETHER EXCHANGING "MESSAGES" IN ORDER TO SYNCHRONIZE THEIR ACTIVITY.
IN PRACTICE, IT MAY BE EASIER TO HAVE A SUPERVISOR PROCESSOR THAT COORDINATES THE REQUESTS.

PARALLEL STREAMS OF DATA ACQUISITION ↓ ARE OF COURSE THE EASIEST MEAN TO IMPROVE SYSTEM EFFICIENCY.

COMMUNICATION IS A BASIC REQUIREMENT:

- THROUGH DATAWAYS (BUSES) ↔
- SHARED EXTERNAL MEMORIES ←
- DPM (DUAL PORT MEMORIES) ←
- INTERRUPTS

SYNCHRONOUS COMMUNICATION IMPLIES THAT THE SENDER AND THE RECEIVER HAVE AGREED IN ADVANCE AS TO TIMING. EACH UNIT SHARES A COMMON CLOCK OR IT HAS SEPARATE CLOCKS WHICH ARE SYNCHRONIZED (TO WITHIN A SPECIFIC TOLERANCE)

ASYNCHRONOUS COMMUNICATION IMPLIES THAT NO SUCH COMMON TIME IS AVAILABLE

THEY FALL INTO TWO CATEGORIES:

ONE-WAY PROTOCOLS ARE EITHER SOURCE OR DESTINATION CONTROLLED. TIMING IS IMPLICIT AND NO ACKNOWLEDGEMENT OF DATA RECEPTION IS GIVEN. THE MAJOR PROBLEM IS INSURING AN APPROPRIATE DELAY SO THAT DATA IS VALID.

REQUEST/ACKNOWLEDGE PROTOCOLS DEMAND AN ACK FROM THE RECEIVER. AND IN SOME CASES A NEGATIVE ACKNOW. (NACK) TO INDICATE AN ERROR.

... TYPES OF SYSTEMS WE ARE INTERESTED IN ... THE ... CALLED, IN THE CLASSIFICATION SCHEME DUE TO FLYNN,

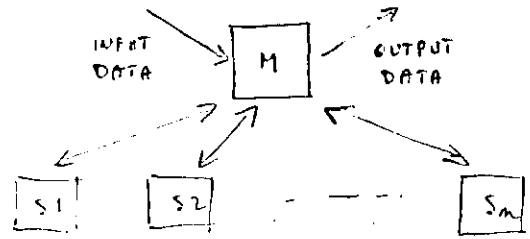
MIMD, I.E. MULTIPLE INSTRUCTIONS, MULTIPLE DATA

WITHIN THIS CATEGORY, WE CAN IDENTIFY DIFFERENT TYPES OF SYSTEMS, ACCORDING TO THE DEGREE OF PHYSICAL AND LOGICAL COUPLING BETWEEN PROCESSORS

1. LOOSE COUPLING
2. MODERATE COUPLING
3. TIGHT COUPLING

ALTHOUGH IT CAN BE HARD TO ASSIGN A GIVEN SYSTEM TO ONE OF THESE CLASSES IN AN UNAMBIGUOUS WAY, ONE CAN GIVE EXAMPLES OF R.T.S. IN CLASSES 2.) AND 3).

A CLASS 2) EXAMPLE COULD BE THE FOLLOWING



A MASTER PROCESSOR TAKES INPUT DATA IN, DISTRIBUTES

" "

IN TURN ; THE SLAVE PERFORMS COMPUTATIONS AND RETURNS THE RESULTS TO THE MASTER.

SLAVES DO NOT TALK TO EACH OTHER.

THIS TYPE OF ARCHITECTURE IS VERY EFFICIENT FOR SYSTEMS WHERE THE LOAD IS VERY HEAVY BUT HOMOGENOUS.

INSTEAD OF COMPUTATION, THE WORK LOAD COULD BE PROCESS CONTROL AND /OR DATA ACQUISITION.

NORMAL R.T.S. TEND TO FALL IN CLASS 3, HENCE THE INTERCONNECTION TOPOLOGY IS IMPORTANT

⟼

THE BASIC BLOCKS THAT MAKE UP OUR SYSTEM ARE:

PROCESSORS — MEMORIES — PERIPHERALS

MANY DIFFERENT CONNECTION SCHEMES CAN BE PROPOSED AND IT IS DIFFICULT TO GIVE A QUANTITATIVE ASSESMENT OF THEIR RELATIVE MERITS IN AN ABSTRACT AND GENERAL WAY.

WE CAN TRY, HOWEVER, TO INDICATE THE CRITERIA WHICH SHOULD BE USED IN EVALUATING THE PERFORMANCE AND MAKING A CHOICE:

-. PERFORMANCE BOTTLENECK.

WHAT IS THE POINT THAT LIMITS THE SYSTEM PERFORMANCE?

CAN WE DUPLICATE OR EXTEND THIS ELEMENT

(F.I. BUS BANDWITH, MEMORY SIZE ...)

2. MODULARITY

WHAT IS THE COST, RELATIVE TO THE TOTAL COST, OF INCREMENTING THE SYSTEM BY ONE UNIT?

OPTIMAL SITUATION IF THIS IS JUST THE COST OF THAT UNIT.

3. FAULT TOLERANCE AND RECONFIGURABILITY

IF ONE COMPONENT FAILS, IS IT CONCEIVABLE TO RECONFIGURE THE SYSTEM BY ACCEPTING A LOWER PERFORMANCE

4 - INTERCONNECTION COMPLEXITY

HOW COMPLICATED IS IT TO ROUTE DATA FROM ONE ELEMENT TO ANOTHER, IN TERMS OF SOFTWARE DECISIONS?

THE PHYSICAL MEAN OF REALIZING THIS CONNECTION IS USING TWO ELEMENTS:

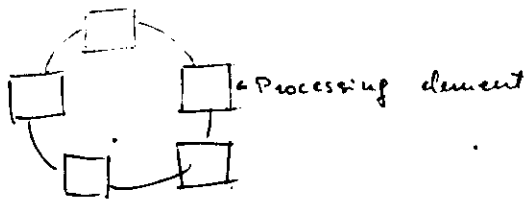
BUS

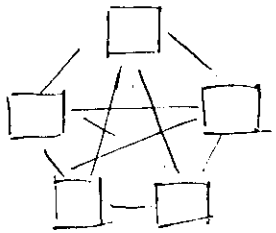
SWITCH

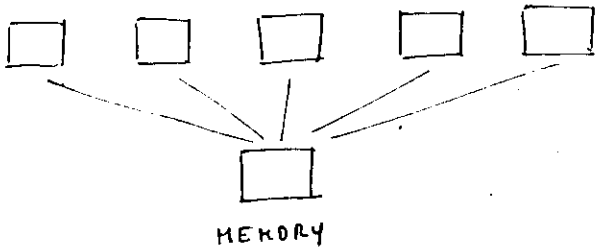
A BUS IS A PHYSICAL PATH ON WHICH MESSAGES ARE TRANSFERRED

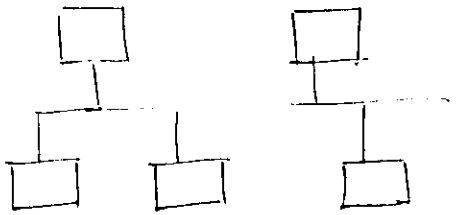
A SWITCH IS A MEAN TO SELECT ONE PARTICULAR PATH.

EXAMPLES OF CONNECTION TOPOLOGIES:



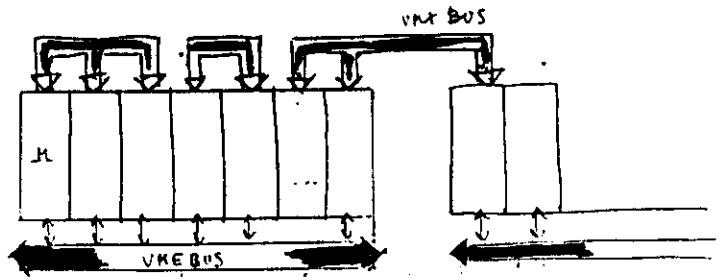






GLOBAL BUS.

AN INTERESTING CLASS OF SYSTEMS IS OFFERED E.G. BY THE VME/VMX BUS STANDARD:



HERE THE SYSTEM ELEMENTS SHARE THE GLOBAL BUS (A CHASSIS BACKPLANE); GROUPS OF ELEMENTS CAN HAVE ALSO AN ALTERNATIVE PATH FOR COMMUNICATION

A BUS ARBITRATION MODULE, INSERTED IN ONE OF THE PROCESSORS, COORDINATES SIMULTANEOUS REQUESTS ACCORDING TO SEVERAL DIFFERENT ALLOCATION SCHEMES

A TIMEOUT MODULE GENERATES A BERR (BUS-ERROR) SIGNAL IF THE TRANSACTION IS NOT PROPERLY TERMINATED BY A DTACK.

A BROADCAST MODE IS AVAILABLE AND AN ADDRESS INCREMENT MODE FOR BLOCK TRANSFERS.

PROCESSORS FOR MULTIPROCESSING

EARLY MICROPROCESSORS, MICRO OR µC CLASS, WERE NOT DESIGNED FOR MULTIPLE PROCESSOR APPLICATION

CAPABILITIES THAT ARE PARTICULARLY USEFUL ARE

- IMPLEMENTATION OF INDIVISIBLE OPERATIONS
- SOPHISTICATED ADDRESSING MODES FOR STACKS AND QUEUES MANIPULATION (FOR SEMAPHORE AND FOR INTERRUPT-CONNECTED CONTEXT SWITCHING)

SAVING-RESTORING REGISTERS IS THE FACTOR THAT MOST CONTRIBUTES TO SYSTEM OVERHEAD.
MORE ADVANCED ARCHITECTURES (MORE REGISTERS!) GIVE ORIGIN TO LARGER DELAYS.

GREAT IMPORTANCE HAS THE HARDWARE CAPABILITY OF ARBITRATING GLOBAL RESOURCES REQUESTS.

ARBITRATION IMPLIES THAT, ON SIMULTANEOUS REQUESTS, ONE OR MORE PROCESSORS MUST WAIT ⟶ THE PROCESSOR MUST HAVE A WAIT STATE

HARDWARE HANDLING OF MULTIPLE PRIORITY INTERRUPT REQUESTS IS IMPORTANT. THIS SHOULD INCLUDE THE CAPABILITY OF DYNAMICALLY ALTER PRIORITIES AND OF MASKING OFF INTERRUPTS

A NON-MASKABLE INTERRUPT LEVEL IS USEFUL

LANGUAGES (AND OTHER BASIC TOOLS)

a) ASSEMBLER, AS ALWAYS.

PROBLEM: AN ASSEMBLY LANGUAGE PROGRAM TENDS TO REFLECT VERY MUCH THE PERSONALITY OF ITS AUTHOR ⟶ DIFFICULT TO UNDERSTAND FOR OTHERS!

b) RT-VERSIONS OF „HIGH LEVEL" LANGUAGES:

FORTRAN, BASIC, PASCAL

c) LANGUAGES THAT ALLOW EXPLICIT HANDLING OF CONCURRENCY:

CONCURRENT PASCAL, MODULA 2

2

CROSS-ASSEMBLERS
COMPILERS
LINKERS

HIGH-LEVEL LANGUAGE LIBRARIES PREPARED AND STORED ON EPROM

2

SMALL MONITOR SYSTEMS ON EPROMS.

2

IF POSSIBLE, NO OPERATING SYSTEM!

DEBUGGING : SIMILAR CONSIDERATIONS HOLD AS IN THE CASE OF THE UNIPROCESSOR SYSTEM, WITH THE EXTRA PROBLEM THAT THE VIRTUAL CONCURRENCY IS NOW A REAL ONE,

DEBUGGING MUST PROCEED IN SOLID STEPS FROM THE INTERNAL (HARDWARE) SYSTEM STRUCTURE TO THE EXTERNAL ONE (MULTIPROCESSOR ORGANIZATION)

A RING BUFFER, RECORDING THE LAST N TRANSACTIONS CAN BE VERY USEFUL.

WATCHDOG : A PROCESSOR, WHEN ALIVE, INCREMENTS A LOCATION. AN INDEPENDENT PROCESSOR MONITORS THE STATE OF THIS LOCATION. IF IT DOES NOT CHANGE, ERROR MESSAGE.

STATE DIAGRAMS.

WHEN THE ORGANIZATION OF THE MULTIPROCESSOR SYSTEM BECOMES COMPLEX, IT MAY BECOME NECESSARY TO GIVE A MORE FORMAL AND RIGOROUS REPRESENTATION OF THE INTERACTIONS BETWEEN THE VARIOUS ACTIVE ELEMENTS (PROCESSORS).

A STATE DIAGRAM CAN REPRESENT A USEFUL TECHNIQUE.

IN PRACTICAL CASES, IT IS RARELY POSSIBLE TO GIVE A REPRESENTATION OF THE ENTIRE SYSTEM, BUT IT CAN HELP TO SORT OUT TOUCHY POINTS.

TO ILLUSTRATE THE CONCEPT, CONSIDER A SERIAL ADDER AS A SYSTEM THAT CAN EXIST IN TWO STATES:

$C \equiv$ CARRY SET

$\overline{C} \equiv$ CARRY CLEAR

TWO INPUTS (THE TWO BITS OF THE ADDENDS)

ONE OUTPUT (ONE BIT OF THE RESULT)

THE POSSIBLE COMBINATIONS OF THE INPUTS ARE OF COURSE 4 ; EACH WILL PRODUCE AN OUTPUT AND IT CAN PRODUCE A STATE CHANGE ; THE FOLLOWING TABLE CAN SUMMARIZE THE BEHAVIOUR OF THE ADDER:

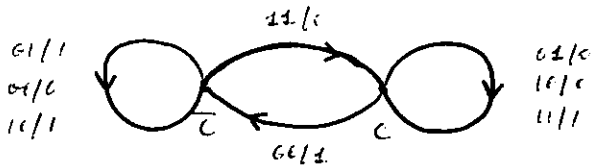
INPUT

CURRENT STATE	00	01	10	11
$\bar{c}$	$\bar{c}/0$	$\bar{c}/1$	$\bar{c}/1$	$c/0$
c	$\bar{c}/1$	$c/0$	$c/0$	$c/1$

$- / -$ ✓

↑ new state

AN EQUIVALENT GRAPHICAL REPRESENTATION IS THE FOLLOWING:



output ↓

$- / -$

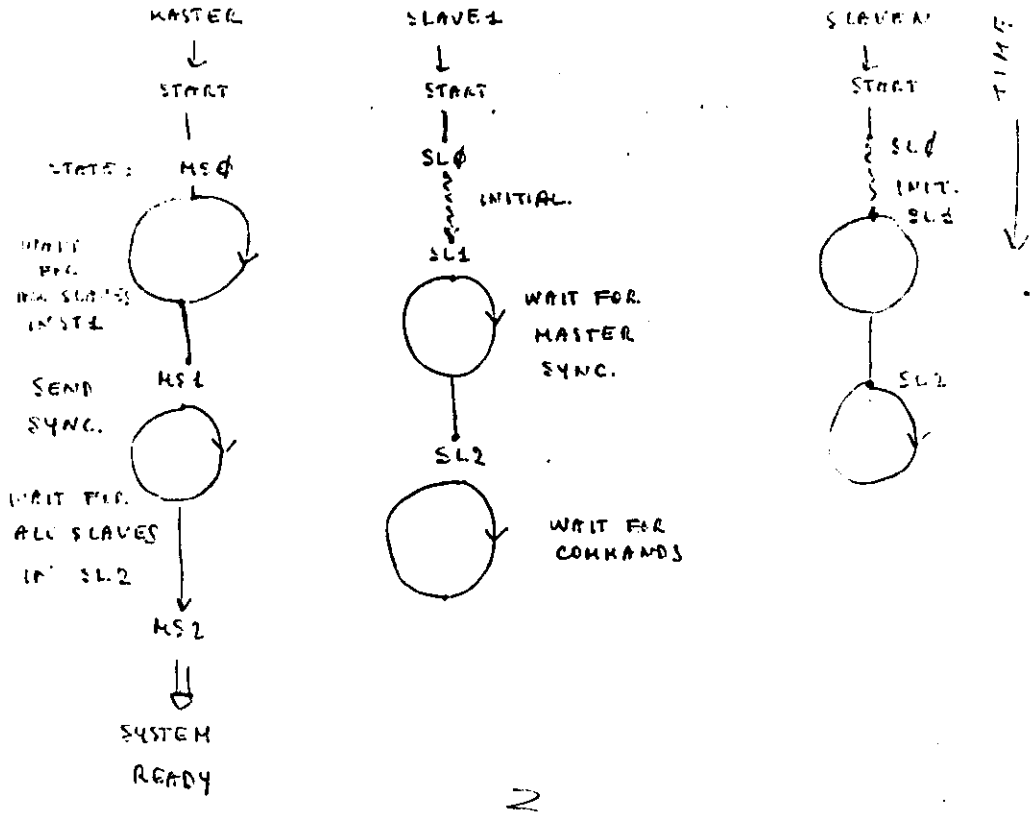
↑ input

SYNCHRONIZATION.

... : 1 MASTER PROC.

N SLAVE PROC.

PROBLEM : GET SYSTEM INITIALIZED AND SYNCHRONIZED.



FORMAL TREATMENT OF THE CONCURRENCY PROBLEM:

"A [illegible] HAVING THE PROPERTY THAT THE INTERCONNECTION NETWORK OF DEDICATED FUNCTION PROCESSORS WHICH MAKE UP THE SYSTEM IS GENERATED FROM A HIGH LEVEL DEFINITION OF DESIRED SYSTEM OPERATIONS CALLED A "CONTROL GRAMMAR". "

THE MULTIPLE PROCESSOR ARCHITECTURE GENERATED IS NOT UNIQUE: SELECTION OF A CANDIDATE MACHINE CAN BE MADE BY APPLYING PERFORMANCE CRITERIA.

THE IDEA IS THAT THE OPERATION OF A PROCESSOR, UPON RECEIPT OF A COMMAND, IS A FUNCTION OF AN INTERNAL STATE AS WELL AS OF THE RECEIVED COMMAND.

PROPERTIES OF THE ARCHITECTURE, SUCH AS DEADLOCK RESOLUTION, AND PERFORMANCE, CAN BE ANALYSED IN A RIGOROUS WAY.