cii COMPAGNIE INTERNATIONALE POUR L'INFORMATIQUE

MITRA 125

REFERENCE MANUAL

VOLUME 2

DECEMBER 1975

N° 5419/U1/EN

SUMMARY

TABLE OF FIGURES

I - GENERAL

I-1. INTRODUCTION

MITRA 125 has two main, distinct program classes differentiated according to their execution mode :

- Privileged Mode
- User Mode

I-2. PRIVILEGED MODE

Privileged mode is divided into three submodes :

- "Privileged Program" Mode
- "Privileged Subroutine" Mode
- "Supervisor" Mode

In Privileged mode, the entire set of computer instructions can be executed. No program or data protection is possible in this mode ; its utilization must be restricted.

I-3. USER MODE

User mode is divided into two submodes :

- "Program" Mode
- "Subroutine" Mode

In User mode, only instructions that cannot exceed various protection boundaries can be executed. This is the normal applications program operating mode. This operating mode is described in VOLUME I.

II - GENERAL DESCRIPTION OF MITRA 125 IN PRIVILEGED MODE

II-1. INTRODUCTION

MITRA 125 utilization in Privileged mode covers exactly its utilization in User mode (described in Volume I of this manual) but offers additional features described in this chapter.

These additional features are as follows :

- Registers :
  - . the general register set already known in User mode may be modified
  - . base and length registers may be read or modified
- Instructions :
  - . a complementary instruction set may be used
- System Segments :
  - . 2 new segment types are directly accessible :

    Supervisor Segments (Base S)
    Context Segments (Base C)

  Other system segments are not directly accessible (no associated base exists) but the system (micromachine) knows that they exist.

    Queue Element Segments
    Segments Containing "Head And Tail Elements" of a queue
    "Coupling By Suspension" Segments

II-2. PRIVILEGED MODE

II-2.1. Privileged Program Mode

A program written in Privileged Program mode is executable in the entire usable memory space. It has the same type of addressing as a program written in Programmed mode (relative to base G) and permits executing special instructions forbidden in User mode.

Privileged Program mode is characterized by the setting of the following status indicators.

SV = 0	Non-Supervisor Mode
SP = 0	Non-Subroutine Mode
PV = 1	Privileged Mode

Privileged Program mode is the execution mode of the program specific for the task.

A program written in Privileged Program mode has access at any given time

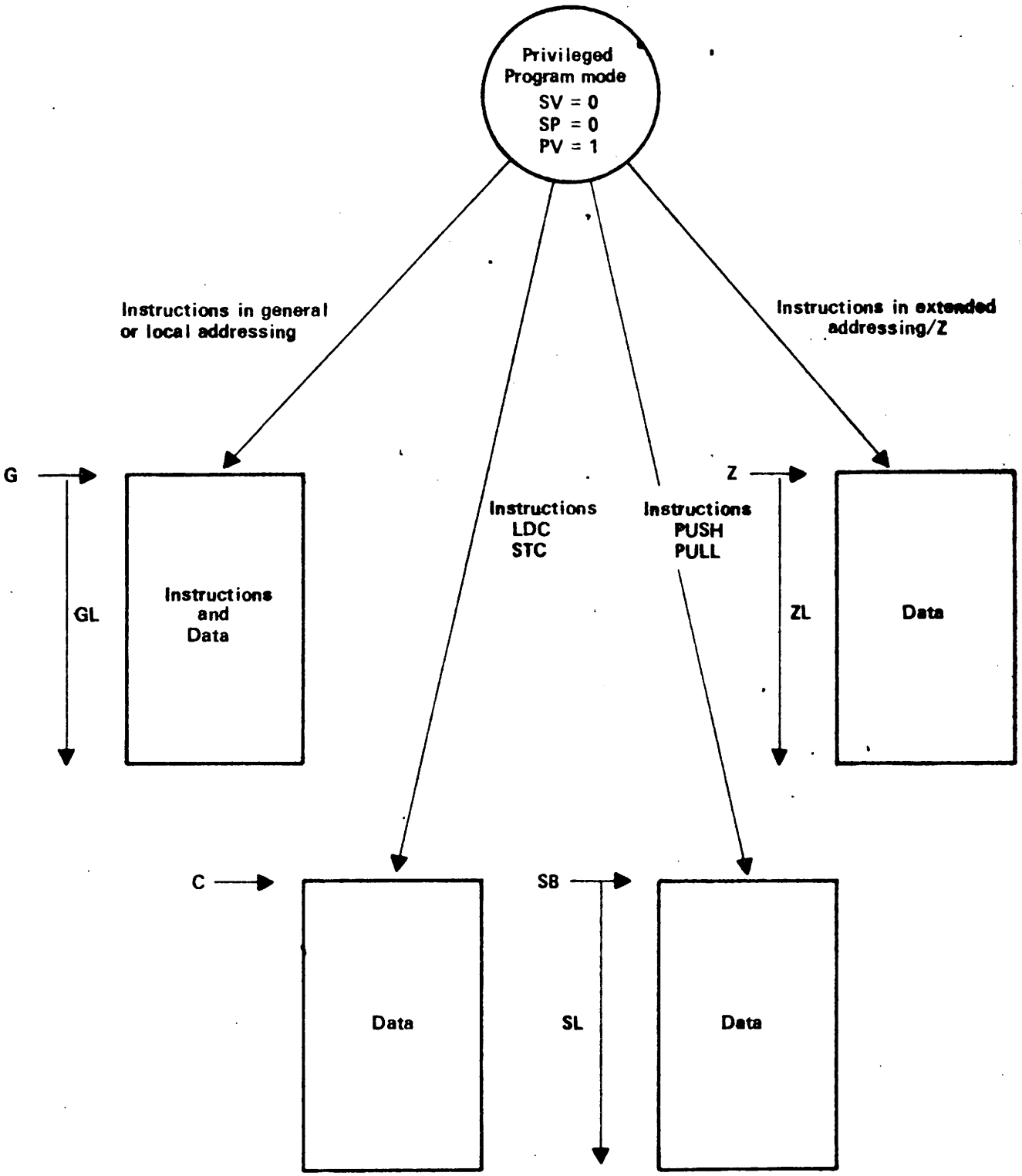
. directly to four segments :

- the program segment defined by the descriptor (G and GL)
- the shared data segment defined by the descriptor (Z, ZL)
- the stack segment defined by the descriptor (SB, SL)
- the context segment defined by the descriptor (C)

For a segment that is not part of the four directly addressable segments, to access this segment, the descriptor of the segment has to be loaded in the appropriate register pair.

The "Privileged Program Mode" topology is given in Figure 2-1.

Figure 2-1. Privileged Program Mode Topology



II-2.2. Privileged Subroutine Mode

A program written in Privileged Subroutine mode is executable in the entire usable memory space. It has the same addressing as a program written in subroutine mode (relative to base Q) and permits executing special instructions forbidden in User mode.

Privileged Subroutine mode is characterized by the setting of the following status indicators.

SV = 0	Non-Supervisor Mode
SP = 1	Subroutine Mode
PV = 1	Privileged Mode

Subroutine mode is the execution mode of an external subroutine which may be shared between several tasks.

A program written in Privileged Subroutine mode has access at any given time

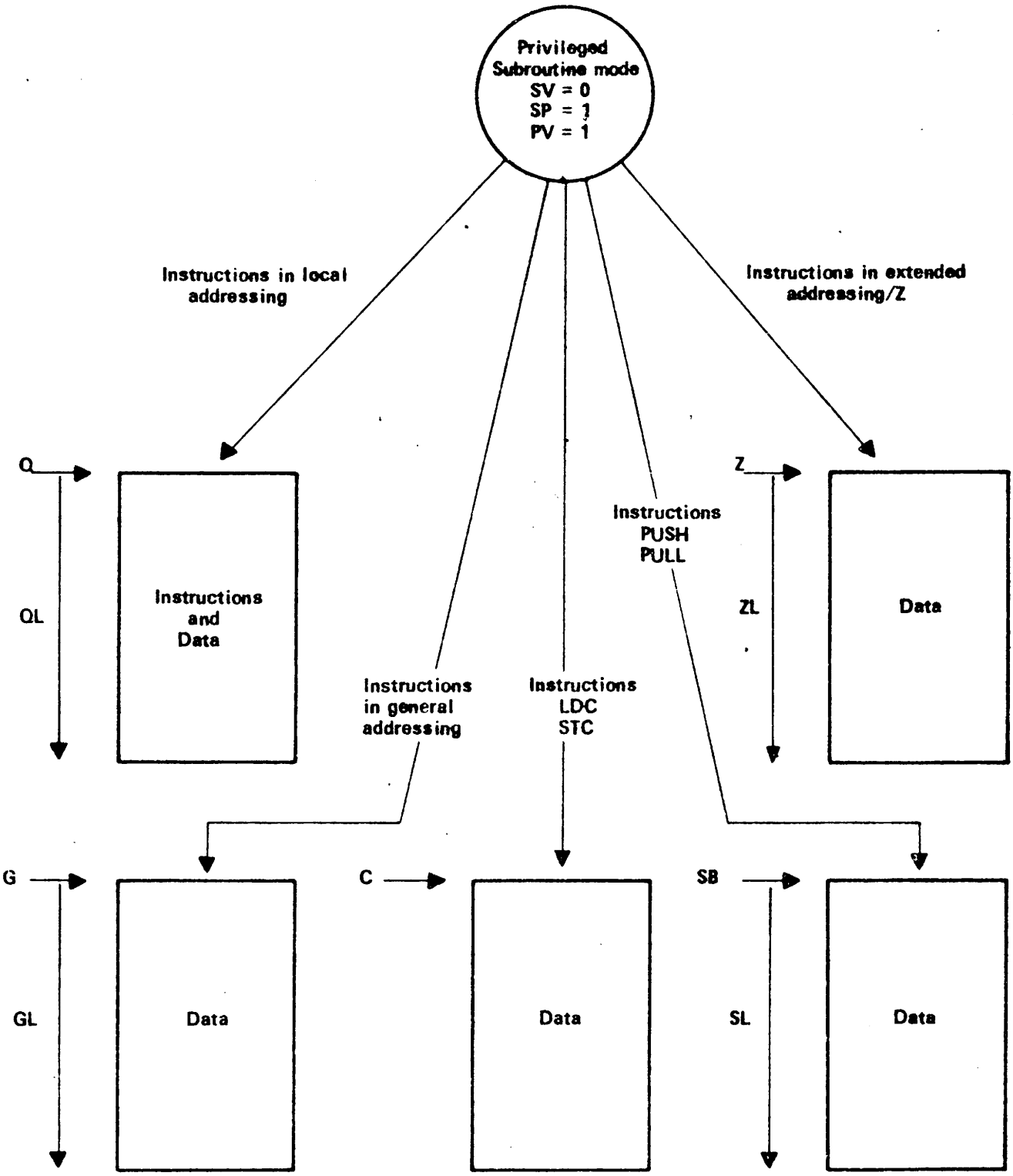
. directly to five segments :

- the program segment defined by the descriptor (G, GL)
- the subroutine segment defined by the descriptor (Q, QL)
- the shared data segment defined by the descriptor (Z, ZL)
- the stack segment defined by the descriptor (SB, SL)
- the context segment defined by the descriptor (C)

For a segment that is not part of the five directly addressable segments, to access this segment, the descriptor of the segment must be loaded in the appropriate register pair.

The "Privileged Subroutine Mode" topology is given in Figure 2-2.

Figure 2-2. Privileged Subroutine Mode Topology



II-2.3. Supervisor Mode

A program written in Supervisor mode is executable only in the first 64,000 memory bytes. It has a special type of addressing (relative to base G) which facilitates access to monitor tasks and to the program segment of the calling task.

It permits execution of special instructions forbidden in User mode.

Supervisor mode is characterized by the setting of the following status indicators.

SV = 1	Supervisor Mode
SP = 0	Non-Subroutine Mode
PV = 1	Privileged Mode

Supervisor mode is the execution mode of monitor modules.

A program written in Supervisor mode has access at any given time

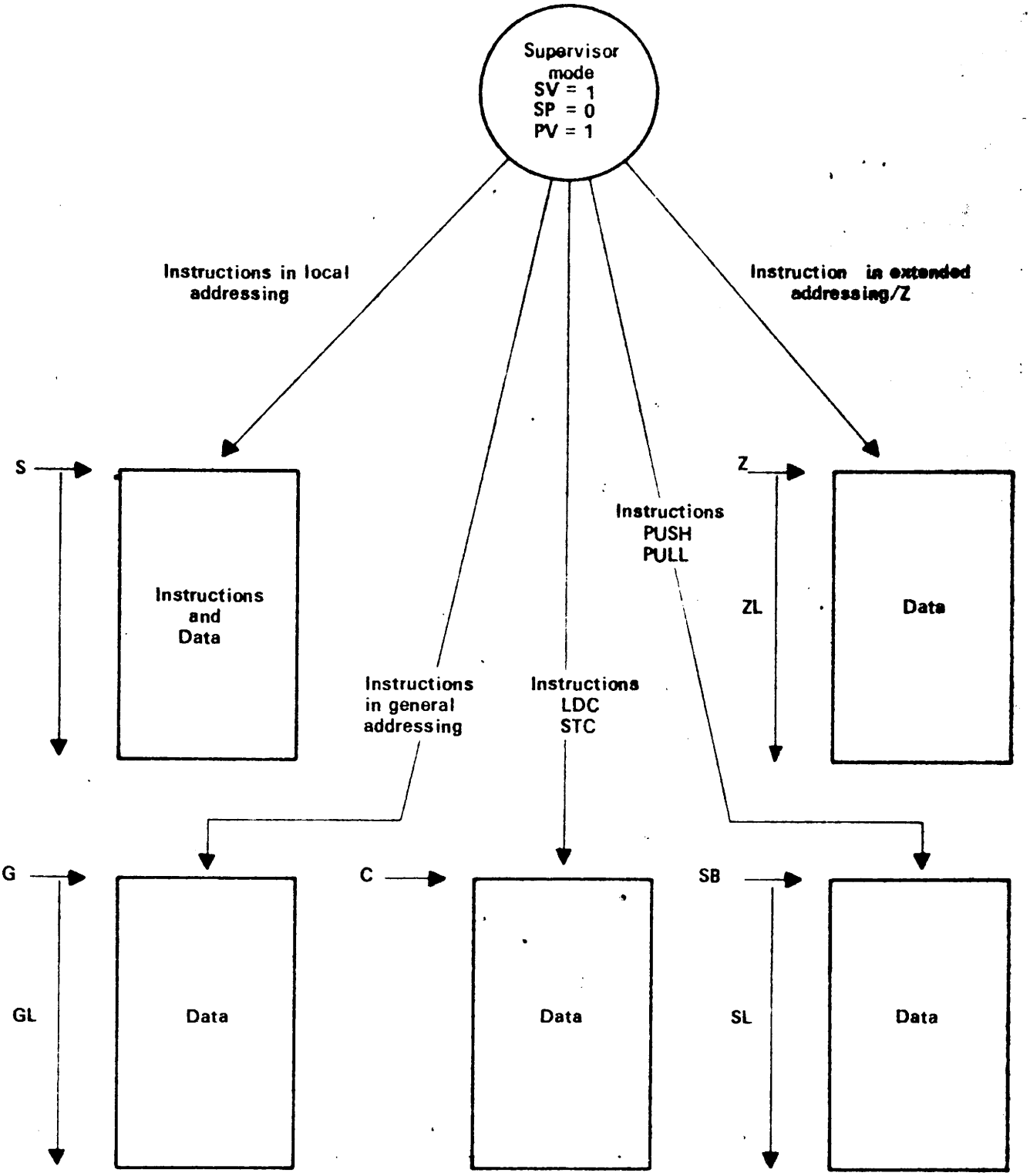
. directly to five segments :

- the program segment defined by the descriptor (G, GL)
- the supervisor segment defined by the descriptor (S)
- the shared data segment defined by the descriptor (Z, ZL)
- the stack segment defined by the descriptor (SB, SL)
- the context segment defined by the descriptor (C)

For a segment that is not part of the five directly addressable segments, to access this segment, the descriptor of the segment must be loaded in the appropriate register pair.

The "Supervisor Mode" topology is given in Figure 2-3.

Figure 2-3. Supervisor Mode Topology



II-2.4. Intermode Transitions (see Figure 2-4)

Transition from one mode to another is performed :

. explicitly by executing instructions :

RTS	Return from missing section
RSV	Return from monitor mode
RTQ	Return from subroutine
STI	Modify the value of indicators

. implicitly by returning from a trap :

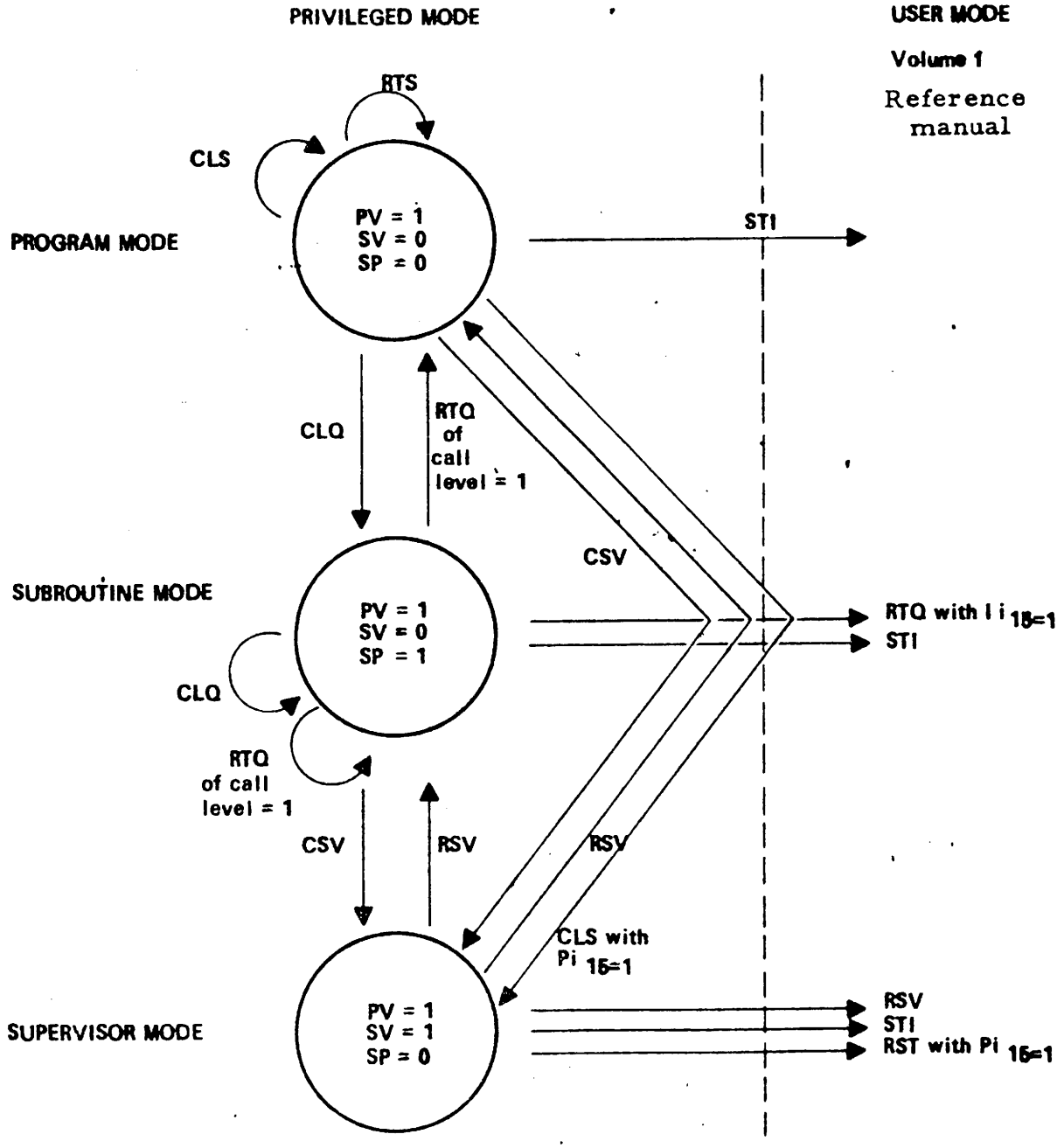


Figure 2-4. Intermode Transitions

II-3. SYSTEM SEGMENTS

II-3.1. Supervisor Segment

II-3.1.1. Definition

The Supervisor segment has a structure identical to a program segment without CDS ; it is constituted by two section types :

. sections which may be called in User mode from the program segment or from the subroutine segment ;
. sections which may be called only in Privileged mode.

A monitor section contains :

. a data subsection (LDS : Local Data Subsection)
. a program subsection (LPS : Local Program Subsection)

An LDS may be common to several LPSs.

A monitor section is called :

. explicitly by instruction CSV ; return by instruction RSV
. implicitly by :
  - a standard trap or a trap specific for the execution (refer to Volume 1)
  - execution of a CLS instruction if the called section is not present in memory (refer to Volume I)

Sections are defined by pointers relative to S and located in a table under S, called PRTS.

Section Zero is the section used for processing traps.

Section One is the section used for loading overlay branches.

Supervisor segment is defined by base register S.

User may access from the supervisor segment by using suitable instructions and types of addressing :

. <u>data</u> :
  - of the supervisor segment itself
  - of the program segment based by G

- of the data segment based by Z
- of the stack segment of the task based by SB
- of the context segment based by C

The "Supervisor Segment" topology is given in Figure 2-5.

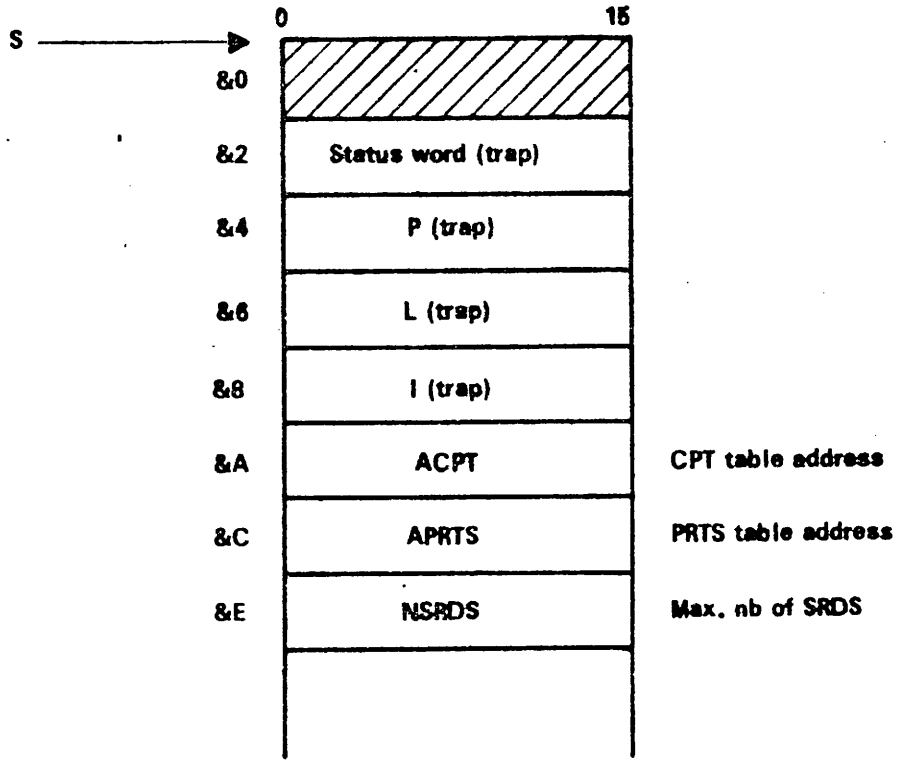
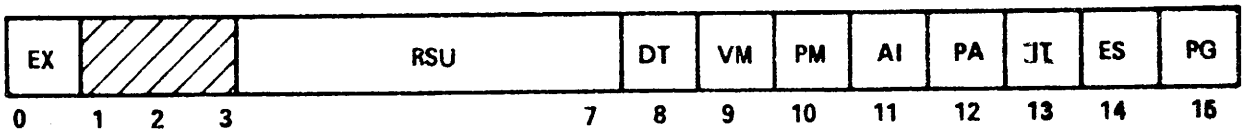


Figure 2-5. Supervisor Segment Topology.

II-3-1-2. Meaning Of The First Supervisor Segment Word

Trap Status Word :



Bit 0 : EX : This indicator at 1 indicates end of instruction EXEC.

Bits 3-7 : RSU : Rank of suspension that caused the trap ; if RSU = 0, trap cause is not a suspension.

Bit 8 : DT : This indicator at 1 indicates that the trap is due to a length overflow - access to a segment with a relative address greater than the segment length.

Bit 9 : VM : This indicator at 1 indicates that the trap is due to a mode violation :

- attempt to execute an instruction in an illegal mode for this instruction
- attempt to call a non-existent supervisor section
- stack overflow during execution of instruction CLQ
- detection that stack empty during execution of instruction RTQ
- attempt to call a non-existent program section.

Bit 10 : PM : This indicator at 1 indicates that the trap is due to a memory protection violation :

- attempt to write in a protected memory cell while protection key indicator PR is at zero or while the console requests total protection mode
- attempt to write under suspension in a protected memory cell while bit 15 of register Tn allocated to suspension n is at zero or while the console requests total protection mode.

Bit 11 : AI : This indicator at 1 indicates that the trap is due to a non-existent address :

- attempt to access a non-existent memory cell.

Bit 12 : PA : This indicator at 1 indicates that the trap is due to a parity error :

- read a memory cell with a parity error
- write a part of a word (including the protection bit) in a memory cell with a parity error.

Bit 13 : II : This indicator at 1 indicates that the trap is due to an illegal instruction :

- attempt to execute an illegal instruction
- end of execution of a BRK instruction

Bit 14 : ES : This indicator at 1 indicates that the trap is due to an Input/Output instruction :

- attempt to dialogue with a missing coupler

Bit 15 : PG : This indicator at 1 indicates that the trap is caused by a program incident and, possibly, by an incident under suspension.

This indicator at 0 indicates that the trap cause is not a program incident.

Word P :

If a trap due to a program incident occurs, word P gives the address of the instruction that has caused the trap.

If a trap is due to an incident under suspension, word P gives the address of the next instruction to be executed in the suspended program.

If an "end of execution of an EXEC instruction" occurs without incident, word P gives the value of register P computed by the instruction designated by EXEC.

If an "end of execution of instruction BRK" occurs, word P gives the address of instruction BRK.

Word L :

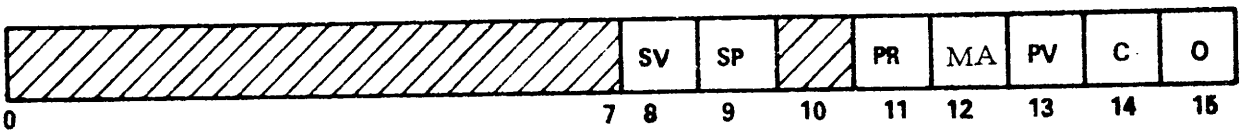
If a trap due to a program incident occurs, word L gives the address of the LDS used at the time of the trap.

If a trap is due to an incident under suspension, word L gives the address of the LDS to be used with the next instruction to be executed in the suspended program.

If an "end of execution of an EXEC instruction" occurs without incident, word L gives the value of register L computed by the instruction designated by EXEC.

If an "end of execution of instruction BRK" occurs, word L gives the value of register L at the time instruction BRK is executed.

Indicator Word :



If a trap due to a program incident occurs, the indicator word gives the status of the indicators before the instruction which has caused the trap is executed.

If a trap is due to an incident under suspension, the indicator word gives the status of the indicators after the instruction which has caused the trap has been executed.

If an "end of execution of an EXEC instruction" occurs without incident, the indicator word gives the status of the indicators after the instruction designated by EXEC is executed.

If an "end of execution of instruction BRK" occurs, the indicator word gives the status of the indicators at the time of the BRK.

Word ACPT :

Word ACPT contains the address relative to S of table CPT (Program Context Table).

Word APRTS :

Word APRTS contains the address relative to S of table PRTS (supervisor program relocation table).

Word NSRDS :

Word NSRDS contains the maximum number of SRDSs (supervisor section relocation double-word). The most significant bit of this word must always be at zero.

II-3.1.3. Table DVT : Description (see Fig. 2-6)

Table DVT (DeVice Table) is located in the supervisor segment above table CPT which is located at address ACPT relative to S.

Table DVT is a 31-word table initialized by software which permits finding the interrupt address as a function of its level.

There is 1 word/interrupt level except for level 0. The word of table DVT associated with a level-n interrupt is located at address ACPT - 2n/S.

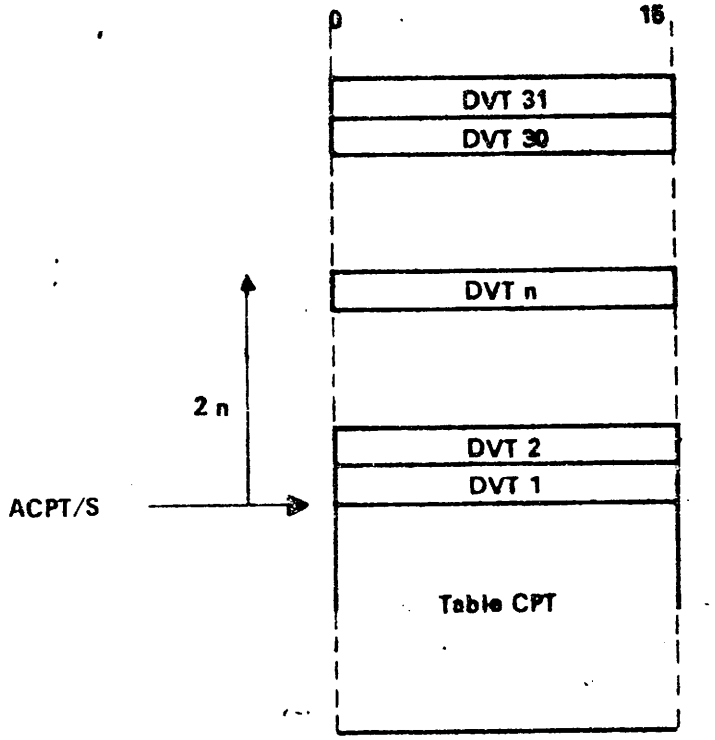


Figure 2-6. DVT Table Topology

DVT Word :

0	1	2	3	4 5 6 7 8 9 10 11 12 13	14 15
D	V	A		IT Number in Group	Group Number

D = Deactivation

V = Enable

A = Arm

II-3.1.4. Table CPT : Description (see Figure 2-7)

Table CPT (Program Context Table) is located in the supervisor segment at address ACPT relative to S.

Table CPT is a 128-word table which contains the 128 bases of the 128 context associated with the 128 interrupts which are divided up into 32 levels of 4 ranks each.

Address ACPT of this table is located in the address word &A of the supervisor segment.

The base associated with the level-n and rank-r interrupt is located at address ACPT + 2n + 64r/S.

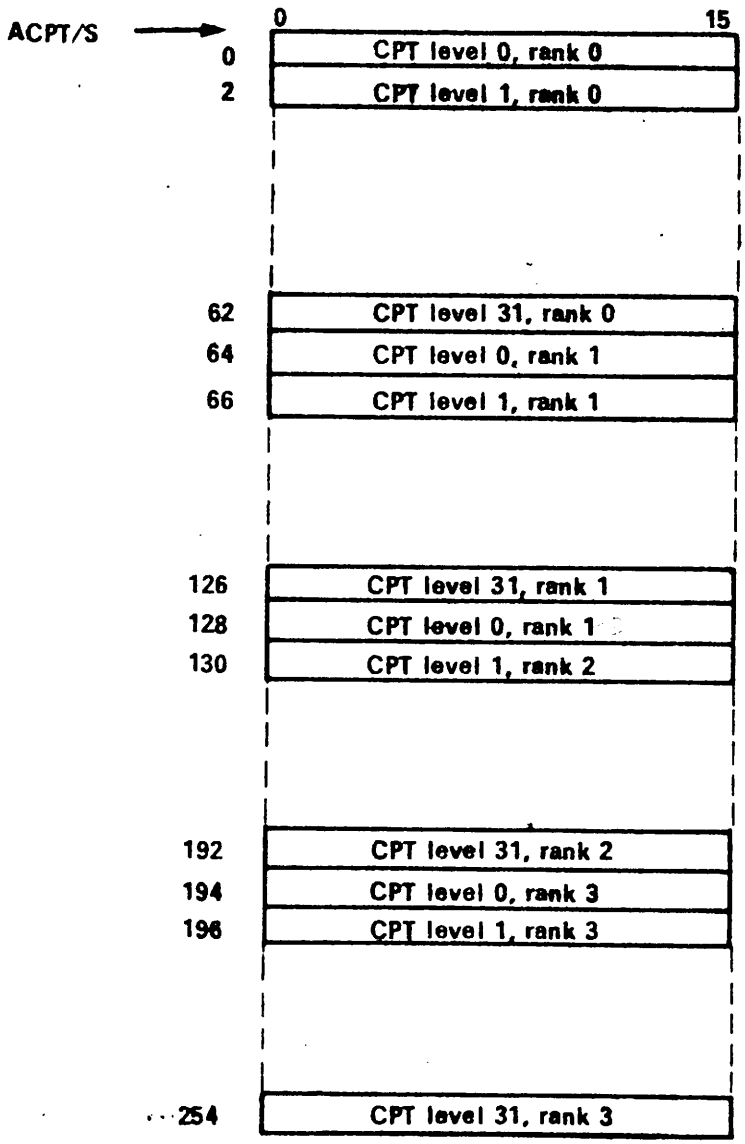


Figure 2-7. OFCPT Table Topology

II-3.1.5. Table PRTS : Description (see Figure 2-8)

Table PRTS (Supervisor Program Relocation Table) is located in the supervisor segment.

Table PRTS is made up of n + 1 double words comprising all SRDSs (Section Relocation Double-word System) of the constituent sections of the system segment.

Maximum number (n + 1) of these SRDSs is contained in the word located at address &E/S of the supervisor segment.

Address APRTS of SRDS of section 1 is in address word &C/S of the supervisor segment.

Address of SRDS of section i is at address APRTS - 4i/S.

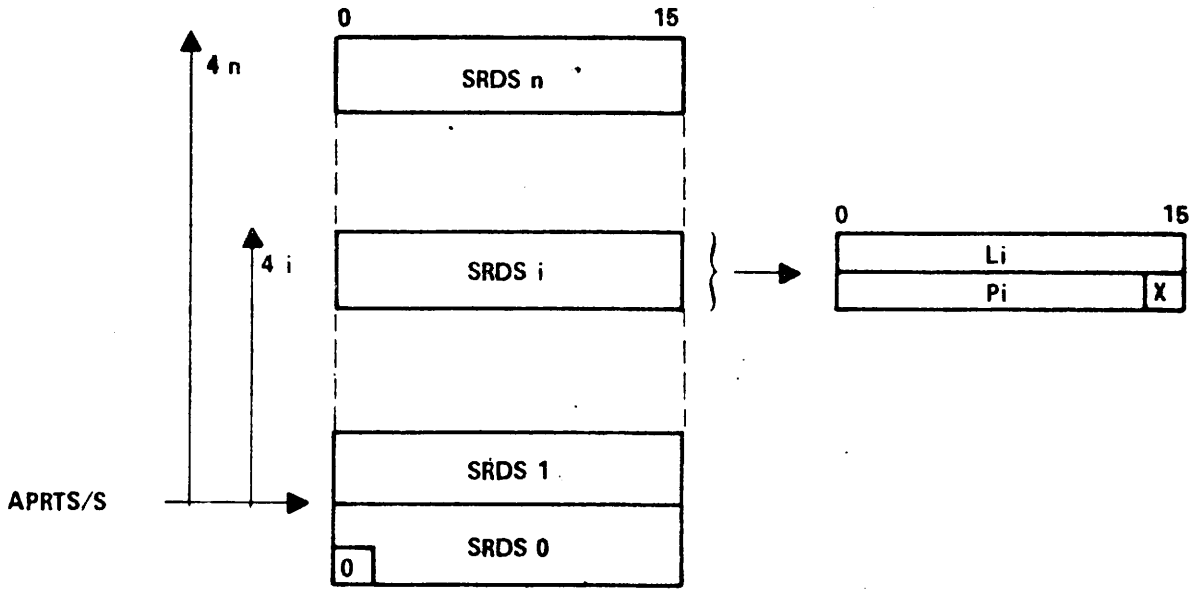


Figure 2-8. PRTS Table Topology

SRDSi contains the double-word which will be loaded in register L and in register P after the execution of the call instruction of section i of the supervisor (CSVi).

SRDS 0 is used only for trap acceptance. Bit 0 of P0 is null.

If bit 15 of Pi equals 1, the calling section must be in Privileged mode ; otherwise, a mode violation trap will occur.

II-3.2. Context Segments (see Figure 2-9)

II-3.2.1. Definition

The context is a segment defined by base C register.

The context segment contains :

- a 10-word switched context (CTX)
- the initial 2-word value of the subroutine segment descriptor (SPSD0)
- a 3-word stack descriptor (PD)
- 3 words (P, L, I) used for instructions EXEC and RTD
- a 1-word descriptor (B:GST) of the table of descriptors of segments common to a task group (GST)
- a 1-word pointer relative to C (T:TST) of the table of descriptors of segments specific for the task (TST)
- word CSN
- task parameter table (TPT)
- table of descriptors of segments specific for the task (TST) - 512 words max.
- a software reserved area

There is a context segment associated with each interrupt, i.e., 128 context segments maximum.

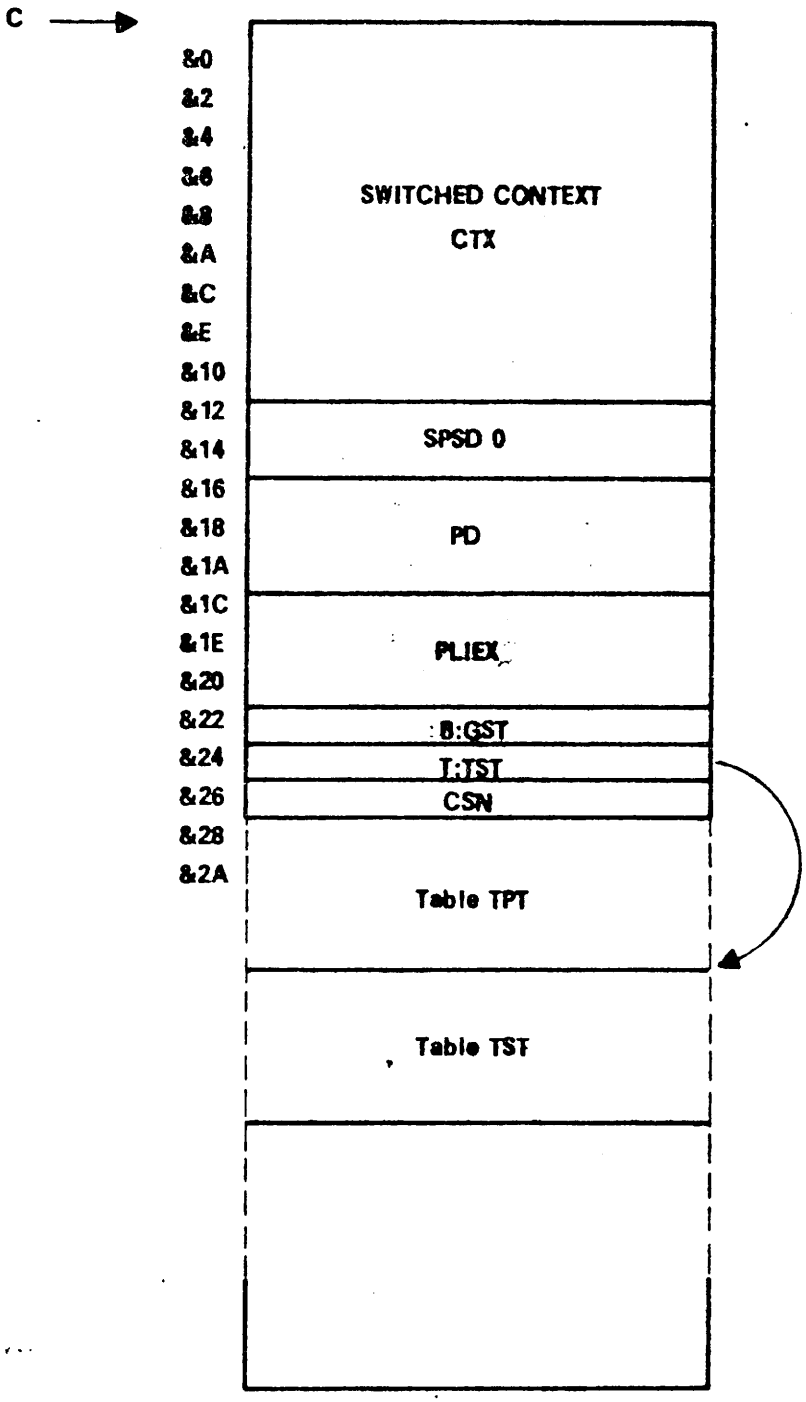


Figure 2-9. Context Segment Topology

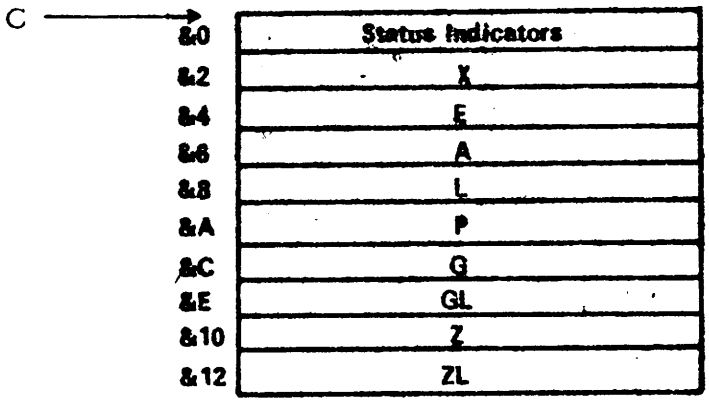
II-3.2.2. Switched Context : Description (CTX)

Definition :

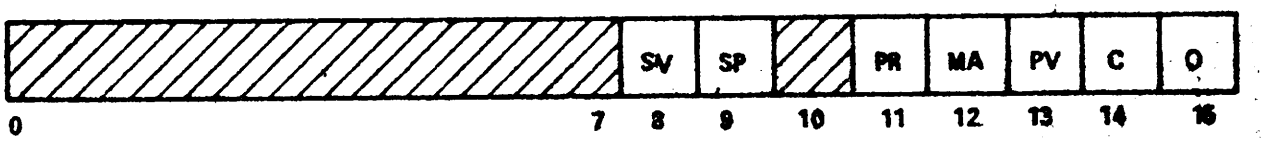
The switched context permits initializing, restarting, and saving the status of a task depending on whether the interrupt level associated with this task is activated or deactivated.

The switched context occupies 10 consecutive words in memory starting from the address pointed by the base C register.

Locations :



Meaning Of Status Indicators :



Bit 8 : SV : Supervisor Mode

This indicator combined with indicator SP modifies the addressing by altering the value given to G'.

When it is set (to 1), indicator SP is forced to 0 and indicator PV is forced to 1.

Bit 9 : SP : Shared Program Mode

This indicator combined with indicator SV modifies the addressing by altering the value given to G'.

Bit 11 : PR : Protection

This indicator at 1 means that the user may access, in

read or write, memory cells protected by a protection bit.

Bit 12 : MA : Mask

This indicator at 1 means that interrupt acceptance is forbidden (except during execution of instructions DIT and XCTX)

Bit 13 : PV : Privileged Mode

This indicator at 1 means that the user may use all the instructions.

Bit 14 : C : Carry

This indicator at 1 means "carry" for the arithmetic type instructions ; for other instructions, its meaning depends on the instruction.

Bit 15 : O : Overflow

This indicator at 1 means "overflow" for arithmetic type instructions ; for other instructions, its meaning depends on the instruction.

X : Index register
E : Accumulator extension
A : Accumulator
L : Local data base
P : Program counter
G : Programmed segment base
GL : Length associated with programmed segment
Z : Data segment base
ZL : Length associated with data segment

General registers (P, L, A, E, X), status indicators, and base and length registers (G, GL, Z, ZL) are automatically saved and restored at each context switching.

Base and length registers (Q, QL) are not saved, but are automatically restored with the contents of the new context at the time of context switching.

Stack descriptor (SB, SL, ST) resides permanently in the context.

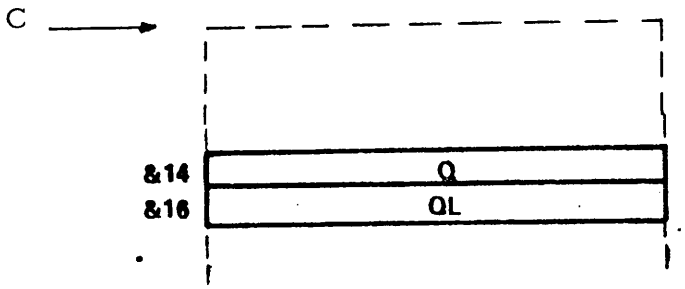
Determination Of Value Given To Base G'

PRIVILEGED MODE	PV	SV	SP	G'
User	1	0	0	= G
Subroutine	1	0	1	= Q
Supervisor	1	1	0	= S

Subroutine Segment Descriptor (SPSD0)

The subroutine segment descriptor (SPSD0) contains the initial descriptor value.

It permits initialization or restoration of base Q and length QL registers at the time of task initialization or resumption. Base Q and length QL registers are not saved in the descriptor at the time of an interrupt.

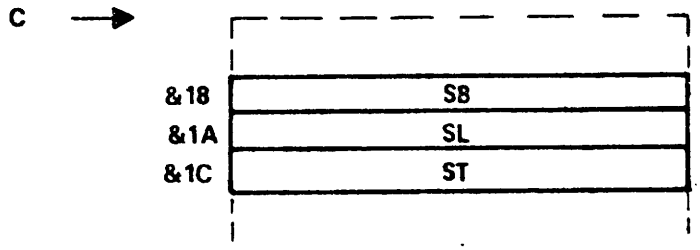


Stack Segment Descriptor (PD)

The stack segment descriptor is made up of three words.

The first two words are the stack segment descriptor associated with the task ; its base and length are respectively SB and SL.

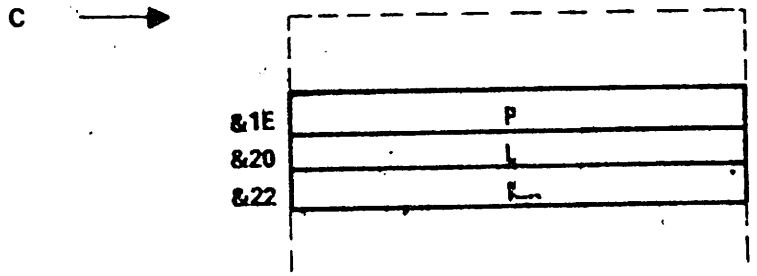
The third word ST permits stack management ; it contains the relative address of the control word of the last stacked element. It must be initialized to zero at the time of stack creation.



These 3 words are used by instructions PUSH (Stack-In), PULL (Stack-Out), CLQ (Call Subroutine), and RTQ (Return From Subroutine).

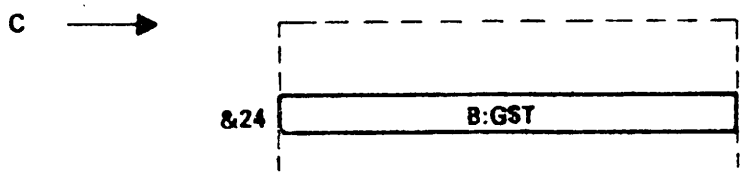
Words P, L, I

The three context words P, L, I are used by instructions EXEC (Execute a designated instruction) and RTD (Return Trap) ; they must be initialized by software before these instructions are executed :



Word B:GST

Word B:GST is the descriptor of the table of descriptors of segments common to a task group (GST) ; it contains the address of table GST. This word is used by instructions LDG (Load base G and length TG) and LDZ (Load base Z and length TZ).



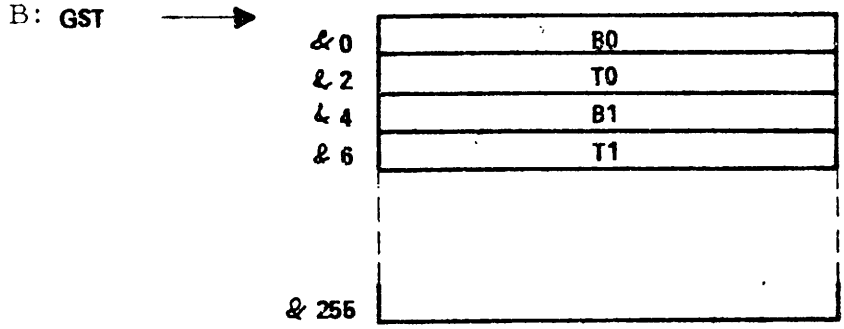
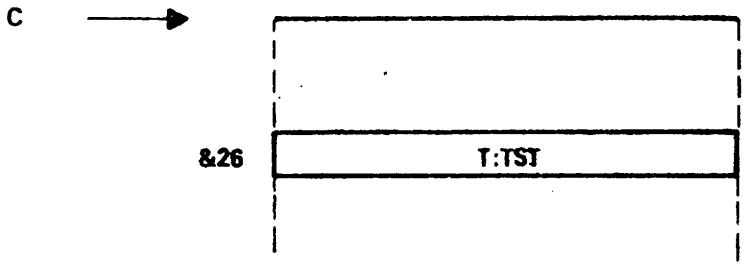


Table GST is the table of descriptors of segments common to a task group. It contains 256 two-word descriptors maximum, i.e., base Bn and length Tn.

Word T:TST

Word T:TST contains the address relative to C of the table of descriptors of segments specific for the task.



Word CSN

This word is modified by all CLS and CLQ instructions executed by the program associated with this context.

Word CSN is significant after execution of a CLS j instruction transformed into CSV = 1 ; it contains then value j + 1.

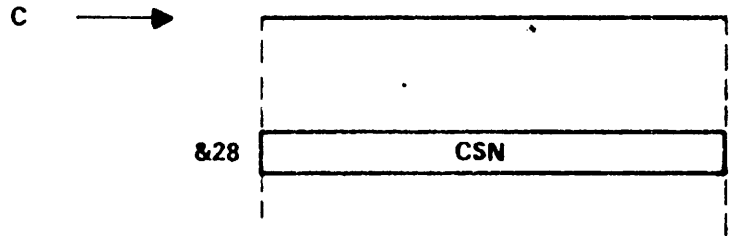


Table TPT

Table TPT is the table of parameters specific for a task :

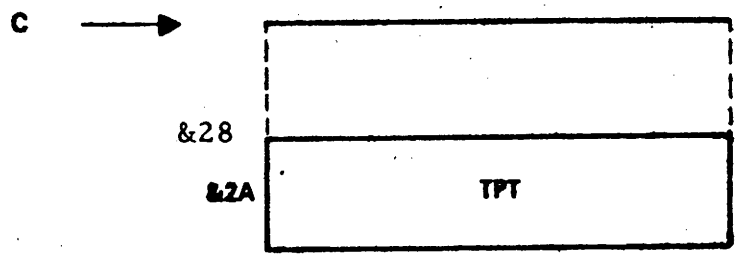
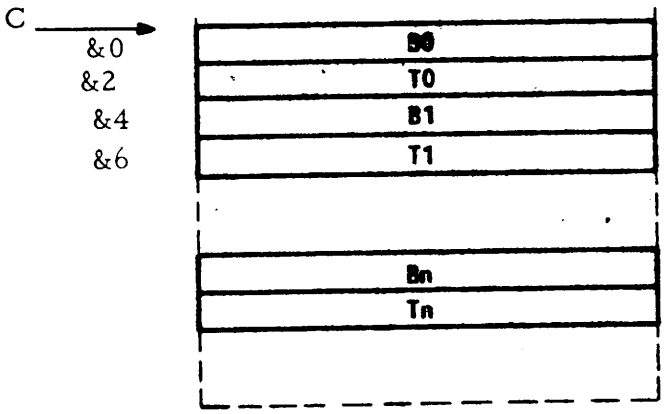


Table TST

Table TST is the table of descriptors of segments specific for a task. It contains 256 two-word descriptors max., i.e. base Bn and length Tn.

This table is used by instructions LDG (Load base G and length GL) and LDZ (Load base Z and length ZL).



II-3.3. Queue Elements Segments (see Figure 2-10)

Elements of a queue QUE (Queue Element) must be in a segment QUS (Queue Segment).

Each element is referenced by its relative address in segment QUS.

Each element comprises 5 control words followed by a message of any length.

Within the segment, elements are chained (upstream and downstream chaining) followed by the element's priority.

Descriptor of segment QUS (Segment Base and Length) is in the first two words of segments HTS (Head/Tail Segment) used to perform main monitor functions (resource management, semaphore management, event management, delay management,...)

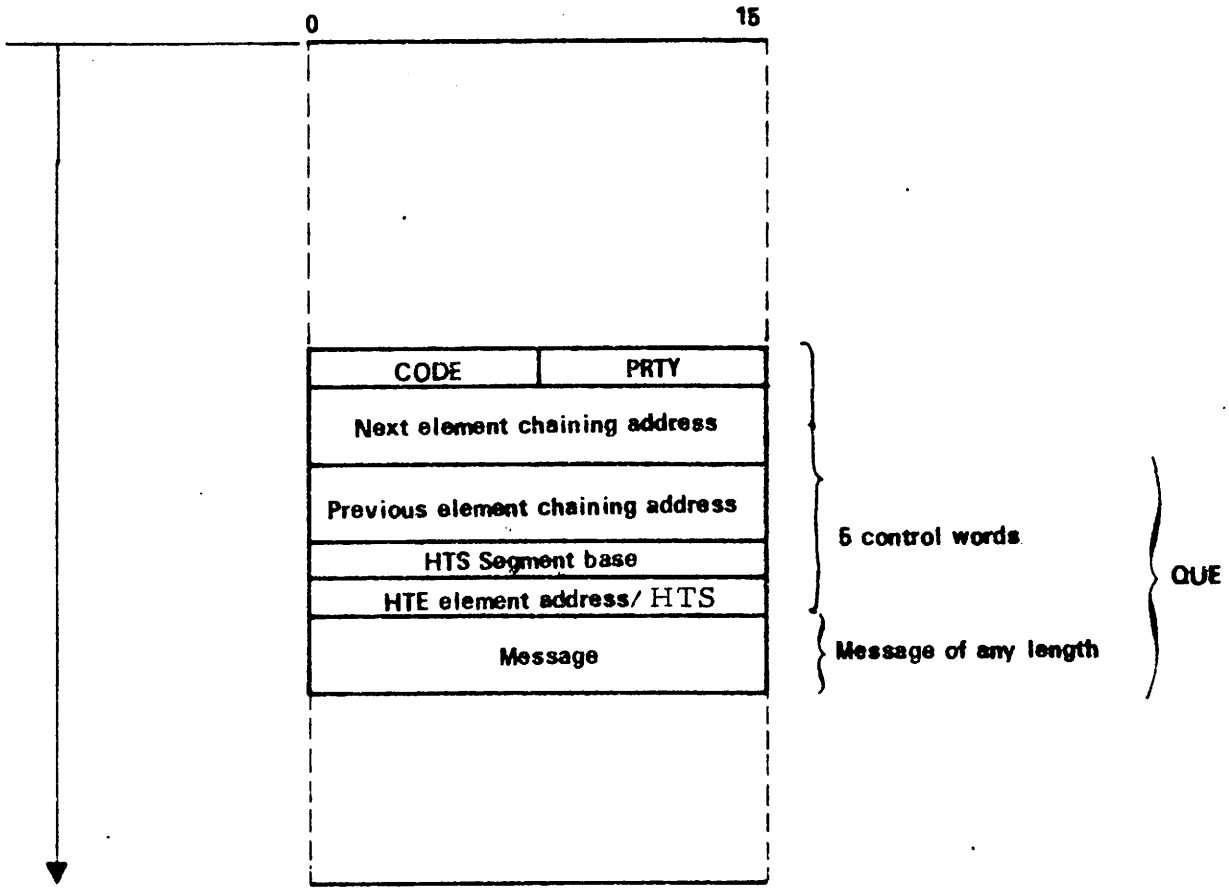


Figure 2-10. Topology Of Queue Elements Segments.

Word 1 :

Code : software reserved byte

PRTY : element priority

= &00 : highest priority

= &FF : lowest priority

Word 2 :

Address, relative to the beginning of segment QUS, of the next element in the queue.

= &00 : if the element is the last in the queue

Word 3 :

Address, relative to the beginning of segment QUS, of the previous element in the queue.

= &00 : if the element is the first in the queue

Word 4 :

Base of corresponding HTS segment containing addresses (relative to the beginning of segment QUS) of "Head and Tail Elements".

= &00 : if the element is not queued

Word 5 :

Address, relative to the beginning of the corresponding HTS segment, of the HTE element containing addresses (relative to the beginning of segment QUS) of "Head and Tail Elements".

Message Words :

Area of any length reserved for software.

Queue Management Instructions

There are five queue management instructions :

INQ : queue-in an element

INQP : queue-in an element with a given priority

TESQ : test first queue element

OUTQ : queue-out first queue element

DELQ : delete a queue element

II-3.4. Segments Containing "Head And Tail Elements" (see figure 2-11)

"Head and Tail Elements" : HTE (Head Tail Element) must be recorded in an HTS segment (Head Tail Segment).

Each HTE element is referenced by its relative address in segment HTS.

Each HTE element comprises two words.

The first two words of segment HTS contain the QUS segment descriptor (Queue Segment) : queue segment base and length.

There is one HTS segment per main monitor function (resource management, semaphore management, event management, delay management, ...)

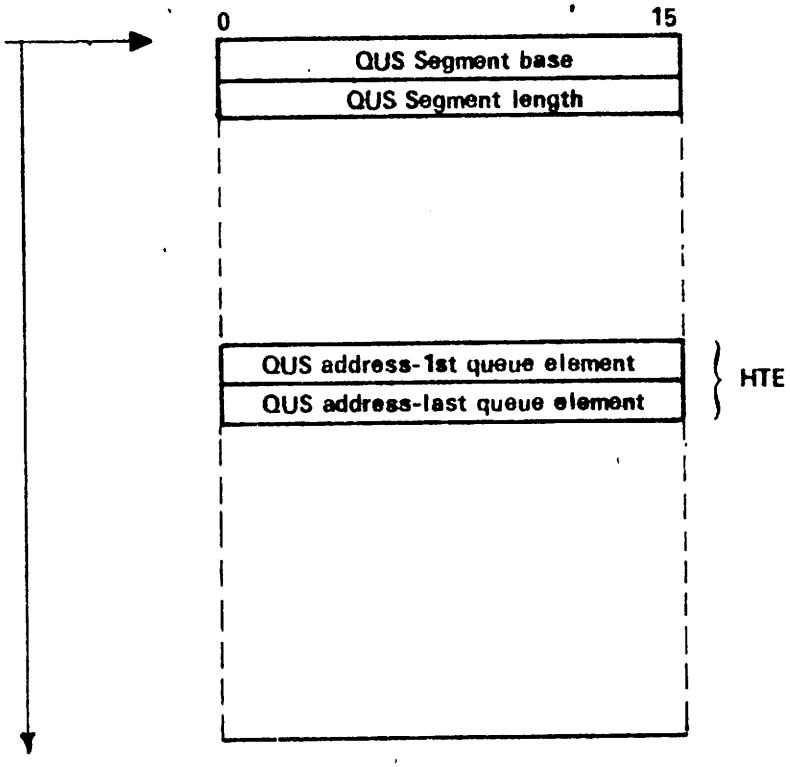


Figure 2-11. Topology Of Segment Containing "Head And Tail Elements"

HTS segments will be addressed via descriptor Z, ZL which will be loaded with the base and length of the HTS segment used.

Word 1 : QUS segment base

Word 2 : QUS segment length

Word 1 of HTE : address (relative to the beginning of the QUS segment)

of the first queue element.

= &00 : if queue is empty

Word 2 of HTE : address (relative to beginning of the QUS segment) of the last queue element.

= &00 : if queue is empty.

II-3.5. "Coupling By Suspension" Segments (see Figure 2-12)

In coupling by suspensions, exchanges between the Input/Output Processor and the Couplers are executed by block.

Up to 26 "coupling by suspension" segments are available.

A "coupling by suspension" segment associated with a rank-n suspension is defined by its base Sn and its length Tn.

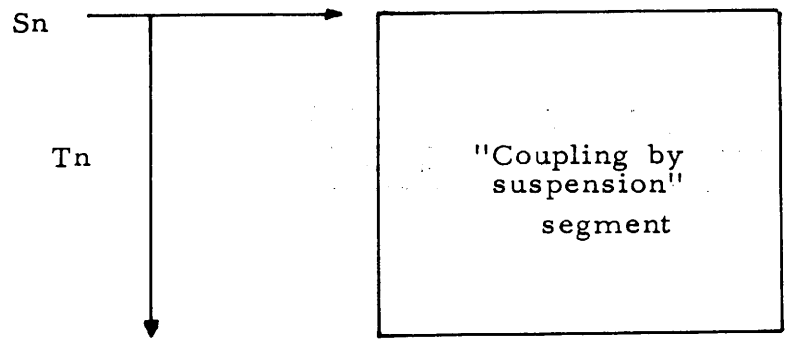


Figure 2-12. Topology Of "Coupling By Suspension" Segment

Base register Sn associated with a rank-n suspension is loaded using instruction STB with N = 62 - 2n as a computed operand.

This base register designates the base used to compute the absolute address of the data buffer.

Length register Tn associated with a rank-n suspension is loaded using instruction STB with N = 63 - 2n as a computed operand.

Only bit 15 of this register is used ; it represents the memory protection key associated with the suspension.

There is no length protection during input/output.

A check is performed with the memory protection lock and the memory protection key associated with the suspension - in case of protection, only read is possible.

II-4. REGISTERS

II-4.1. General Registers

II-4.1.1. Access To General Registers

General registers may be accessed as in User mode. In addition, they may be modified by general instruction STR r.

II-4.1.2. General Registers Description

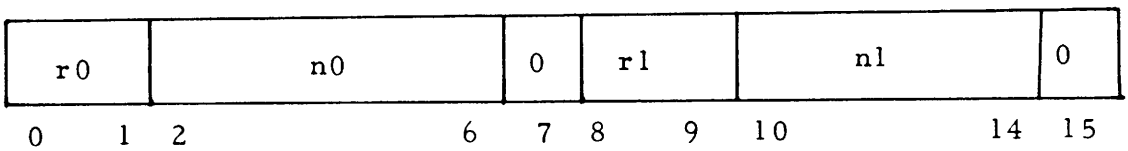
Same description as in User mode with the following differences :

- the contents of registers P an L are relative to S in Supervisor mode ;
- register RIT is the interrupt management register.

The rightmost byte of register RIT contains the rank and level of the current interrupt.

The leftmost byte of register RIT contains the rank and level of the interrupt which was being processed before the current interrupt.

Contents of register RIT are modified by instruction DIT and whenever an interrupt is accepted.



r : interrupt rank
n : interrupt level

The 128 interrupts are divided up into 32 levels (0 through 31) of 4 ranks each (0 through 3) and arranged according to priority ; an interrupt of level n and rank r has a higher priority than an interrupt of level n' and rank r' if and only of :

$$n > n' \text{ or } n = n' \text{ and } r > r'$$

Register RIT Representation

Bits 0 - 1 : rank R0 of interrupt
Bits 2 - 6 : level N0 of interrupt
Bit 7 : zero
Bits 8 - 9 : rank R1 of interrupt
Bits 10 - 14 : level N1 of interrupt
Bit 15 : zero

II-4.2. General Description Of Base And Length Registers

II-4.2.1. Access To Base And Length Registers (see Figure 2 - 13)

Base and length registers are addressable in Privileged mode by instructions LDB and STB or by pressing the console base pushbutton. All registers have 16 bits except for the "coupling by suspension" segment length registers which have only bit 15.

Base register topology is given in Figure 2-13.

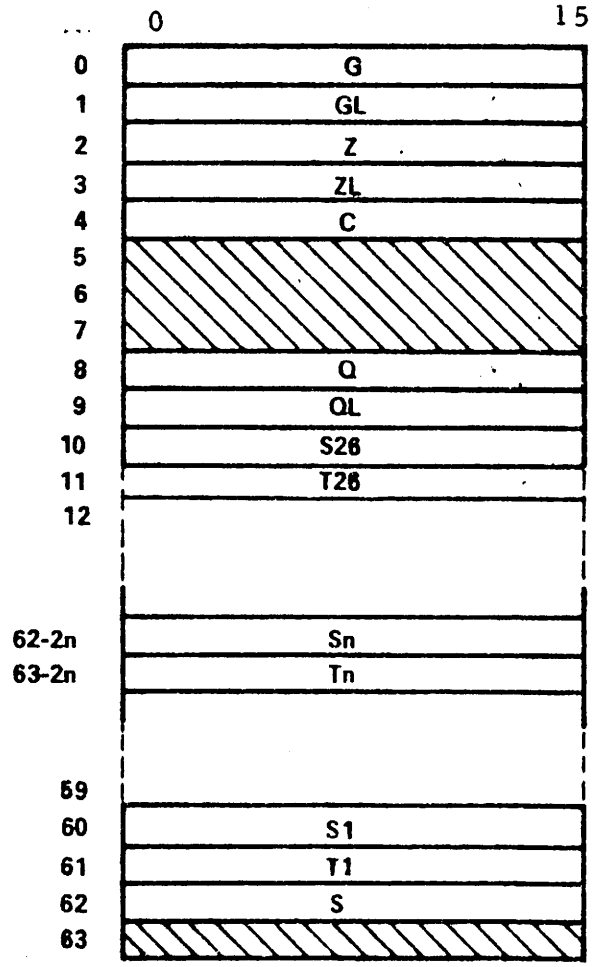


Figure 2-13. Base Registers Topology.

II-4.2.2. Base Registers Description

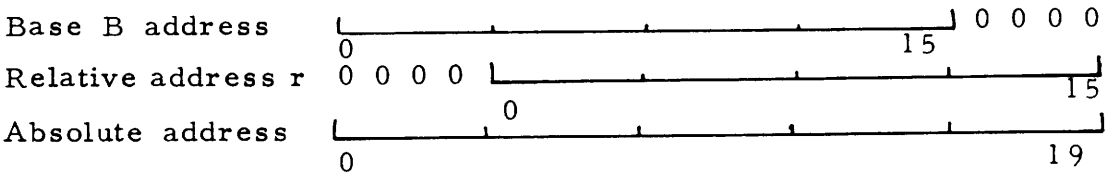
Base registers contain segment descriptors. A segment is a continuous memory area whose start byte-address is a multiple of 16 and which is defined by a segment descriptor.

Base registers contain a work area start address of 64 Kbytes maximum. This address is an octo-word address (multiple of 16 bytes).

When memory is accessed, the selected base B is added to the address r relative to the beginning of the segment in the following manner :

. The 16-bit contents of B are filled in with 4 least significant bits at 0 yielding a 20-bit address.

. This address is added to the 20-bit address obtained by filling in the 16-bit address r with 4 most significant bits at 0.



. The result is therefore a 20-bit address.

The 20-bit absolute address permits directly addressing 1 megabytes.

A length register contains an address relative to the selected base. This value determines the length of the word area under the base.

At each main memory access, a Base/Length pair is selected. By comparing the relative address and the assigned length any access attempt beyond the permitted area may be detected.

A segment descriptor is made up of a base which defines the segment origin and a length which defines the segment length or a protection key for "coupling by suspension" segments.

Program Segment Descriptor (G, GL) :

Address, relative to S, of the beginning of the program segment equals 16 times the contents of base register G ; program segment length equals the contents of length register GL.

Common Data Segment Descriptor (Z, ZL) :

Address, relative to S, of the beginning of common data segment equals 16 times the contents of base register Z ; the common data segment length equals the contents of length register ZL.

Context Segment Descriptor (C) :

Address, relative to S, of the beginning of context segment equals 16 times the contents of base register C. Context segment has no length register.

Subroutine Segment Descriptor (Q, QL) :

Address, relative to S, of the beginning of shared program segment equals 16 times the contents of base register Q ; the shared program segment length equals the contents of length register QL.

"Coupling By Suspension" Segment Descriptors (Sn, Tn) :

Up to 26 "coupling by suspension" segments are available.

Address, relative to S, of the beginning of the segment allocated to suspension n equals 16 times the contents of base register Sn ; the protection key allocated to suspension n is in bit 15 of register Tn.

Supervisor Segment Descriptor (S) :

Absolute address of the beginning of the supervisor segment equals 16 times the contents of base register S.

The supervisor segment has no length register.

III-1. INSTRUCTION ADDRESSING TYPES

Addressing types in Privileged mode are the same as in User mode, except for the following special features :

- G' = base G if Privileged Program mode
  = base Q if Privileged Subroutine mode
  = base S if Supervisor mode
- Some instructions directly address the context segment (base G)

Taking into account these special features, the following should be noted :

- an instruction of a subroutine segment may directly access data belonging to :
  - . the program segment
  - . the subroutine segment
  - . the data segment
  - . the context segment
- an instruction of a program segment may directly access data belonging to :
  - . the program segment
  - . the data segment
  - . the context segment
- an instruction of the supervisor segment may directly access data belonging to :
  - . the supervisor segment
  - . the program segment
  - . the data segment
  - . the context segment

III-2. INSTRUCTION CLASSES

III-2.1. General

Instruction classification specified for User mode remains valid ; new instructions usable only in Privileged mode are :

. in class 1
. in class 1'

III-2.2. Additional Instructions Of Class 1

There are two instructions :

STR	Store Register
XCTX	Exchange Context

These instructions are characterized by :

Bits 4-5 : 11
Bits 6-7 : 10

XCTC is an instruction without an operand.

III-2.3. Additional Instructions Of Class 1'

Additional instructions of this class are stored in families SRG, MCB, COV, and in a new family : family SYS.

Family SYS is characterized by :

Bits 4-5 : 01
Bits 6-7 : 00

III-2.3.1. Family SRG

There is only one additional instruction :

RSV	Return to Supervisor section

This instruction is characterized by :

Bits 10-12 : 001

Bits 13-14 : 10

III-2.3.2. Family MCB

Additional instructions are :

LDC Load Context Element

STC Store Context Element

LDB Load Base

STB Store Base

IN 8 \ IN 0 - 3	1 1 1 0	1 1 1 1
0	LDC	LDB
1	STC	STB

III-2.3.3. Family COV

Instructions of family COV are differentiated by bits 8-15 of the operand computed according to the rules of class 1.

This instruction family permits creating specifically new instructions :

N8-11 \ 12-15	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1 To 1 1 1 1
0 0 0 0	LDZ							
0 0 0 1	LDG							
0 0 1 0	INQ	OUTQ	TESQ	DELQ	INQP	RTD	EXEC	
0 0 1 1	FMUD	FDVD	FADD	FSUD				
0 1 0 0	BRK	BRK	BRK	BRK	BRK	BRK	BRK	BRK
0 1 0 1 To 1 1 1 1								

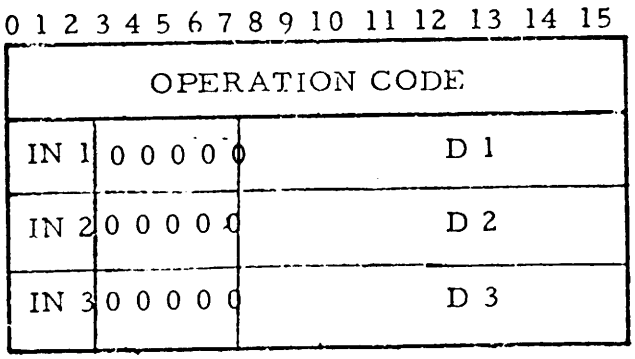
LDZ	Load Base Z and Length TZ by a descriptor
LDG	Load Base G and Length TG by a descriptor
INQ	Queue-in
OUTQ	Output first queue element
TESTQ	Test first queue element
DELQ	Delete queue element
INQP	Queue-in with priority
RTD	Return trap
EXEC	Execute designated instruction
FMUD	Double-precision, floating point multiplication
FDVD	Double-precision, floating point division
FADD	Double-precision, floating point addition
FSUD	Double-precision, floating point subtraction
BRK	Generate illegal instruction trap

Remark : Instructions FMUD, FDVD, FADD, FSUD operate in User mode. They are four-word instructions :

The first word defines the operation mode.

The next three words define the address of operand 1 (IN1, D1), operand 2 (IN2, D2), and the result (IN3, D3).

Each (IN, D) pair defines 8 types of addressing : DL, DG, IL, EL, ILX, ELX, IGX, EGX.



Operands are computed according to the rules of class 0'.

III-2.3.4. Family SYS

The 10 instructions of family SYS are differentiated by bits 12-15 of the operand computed according to the rules of class 1.

Instructions of this family are instructions without an operand.

N 12-13 \ N 14-15	0 0	0 1	1 0	1 1
0 0	CLM	DIT	RD	WD
0 1	SIO	MIO	HIO	CLP
1 0	STM			
1 1				STP

CLM	Clear interrupts
DIT	Deactivate interrupt
RD	Read Direct
WD	Write Direct
SIO	Start I/O
HIO	Halt I/O
AIO	Acknowledge I/O interrupt
CLP	Reset memory protection
STM	Mask interrupts
STP	Set memory protection

IV - STRUCTURE PROCESSING

IV-1. INTRODUCTION

Structure definition and utilization in Privileged mode are exactly the same as for those described in Volume I of this manual for User mode.

However, in Privileged mode, a new type of structures can be defined and used : structures organized as a segment, defined via pseudo-instructions SEGG and SEGZ.

This manual describes how such structures are defined and used.

Note that a structure belonging to the supervisor segment may be accessed only from the supervisor segment.

IV-2. DEFINITION PSEUDO-INSTRUCTIONS OF A STRUCTURE ORGANIZED AS A SEGMENT

IV-2.1. Pseudo-Instruction SEGZ

Function : Declaration of a structure organized as a segment based by Z.

Format

Label	Command	Argument
	SEGZ	N = <segment number> I/D

Result

Pseudo-instruction SEGZ defines the structure as an element belonging to a segment based by Z.

Pseudo-instruction SEGZ must - if it exists - immediately follow pseudo-instruction SIZE.

The presence of pseudo-instruction SEGZ excludes that of pseudo-instruction SEGG.

N = <segment number> is the number of the segment descriptor in GST (Table of descriptors of segments shareable by a task group) or in TST (Table of descriptors of segments specific for a task) of the task context.

Option I indicates that the segment is accessed indirectly ; this segment is in GST.
Option D indicates that the segment is accessed directly ; this segment is in TST.

A structure segment is a data block made up of a certain number of structures.

Example Of A Structure Organized As A Segment Based By Z

```
L1      STRUC
        NAME        N1
        SIZE        S1
        SEGZ
...     ...         ...
        FINST
```

IV-2.2. Pseudo-Instruction SEGG

Function : Declaration of a structure organized as a segment based by G.

Format

Label	Command	Argument
	SEGG	N = < segment number > I/D, BPA = < expression P1 > , BEA = < expression P1 >

Result :

Pseudo-instruction SEGG defines the structure as an element belonging to a segment based by G.

Pseudo-instruction SEGG - if it exists - immediately follows pseudo-instruction SIZE.

The presence of pseudo-instruction SEGG excludes that of pseudo-instruction SEGZ.

N = < segment number > is the number of the segment descriptor in GST (Table of descriptors of segments shareable by a task group) or in TST (Table of descriptors of segments specific for a task) of the task context.

Option I indicates that the segment is accessed indirectly ; this segment is in GST.
Option D indicates that the segment is accessed directly ; this segment is in TST.

Each item of the first segment structure is marked by a pointer ; the area containing these pointers is located at address BPA = < expression P1 >.

First segment structure is located at address BEA = < expression P1 >.

Example Of A Structure Organized As A Segment Based By G

```
L1      STRUC
        NAME        N1
        SIZE        S1
        SEGG
...     ...         ...
        FINST
```

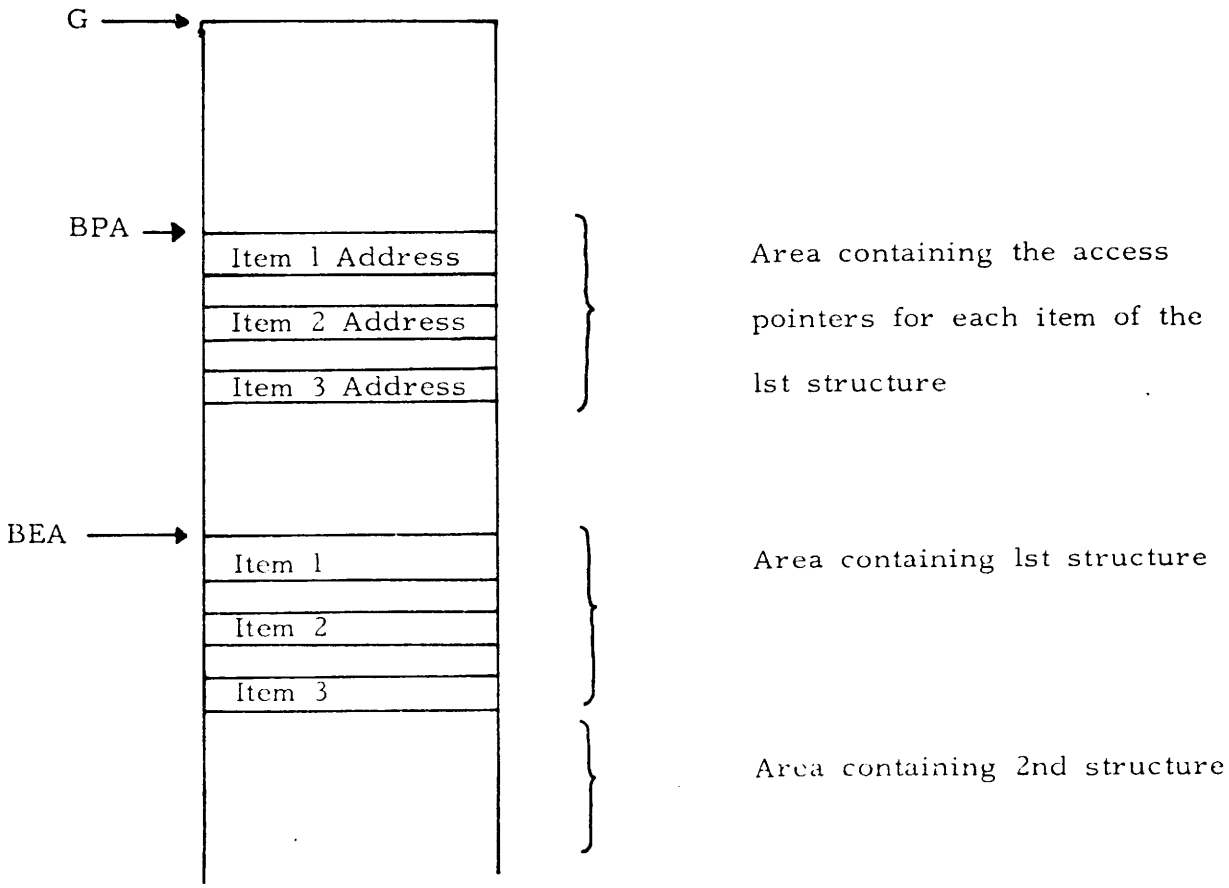


Fig. 4-1 Topology of a segment structure based by G

The various segment structures must be identical.

IV-3. ACCESS TO CONTEXT SEGMENT STRUCTURES

IV-3.1. Access To The Current Task's Context Structure

Context is described by a simple structure called CTX.

When the current task desires to access its context, it uses one of the following two internal macros.

Format

Label	Command	Argument
[< Label >]	LDC	= CTX. < item name >
	or	
[< Label >]	STC	= CTX. < item name >

Generated Code

```
[< Label >]   LDC      = < associated value >
              or
[< Label >]   STC      = < associated value >
```

User Interface

Addressable items are :

I	Status Indicators
X	Register X
E	Register E
A	Register A
L	Local Base L
P	Program Counter P
G	Program Segment Base
GL	Program Segment Length
Z	Data Segment Base
ZL	Data Segment Length
Q	Subroutine Segment Base
QL	Subroutine Segment Length

SB	Stack Segment Base
SL	Stack Segment Length
ST	Stack Segment Top
PEX	Interface EXEC : Program Counter P
LEX	Interface EXEC : Local Base L
IEX	Interface EXEC : Indicators
B:GST	GST Base
T:TST	TST Address relative to C
CSN	Number of last called section

IV-3.2. Access To The Current Task's TPT Structure

Table TPT is described by a simple structure called TPT.

When the current task desires to access its TPT table, it uses one of the following two internal macros.

Format

Label	Command	Argument
[< Label >]	LDC	= TPT. < item name >
or	or	or
[< Label >]	STC	= TPT. < item name >

Generated Code

[< Label >]	LDC	= < associated value >
	or	
[< Label >]	STC	= < associated value >

User Interface

Addressable items are :

TKST	: Task Status
TKSYN	: Task System Name
LOCID	: Local Identification Of Task
T:WKA	: Word Area Address Relative To G
T:BNDA	: Binding Area Address Relative To G

CDSL : CDS Length

SYSBLI : Index Of System Block Accessed By Task

DGNEPL : Name Of Group DOWN For A Root Or Utility Task And Library Number Accessible By Task

STPT : Program Start Point

IV-3.3. Access To The Current Task's TST Structure

Table TST is described by a simple structure called TST.

When the current task desires to access its TST table, it uses one of the following two internal macro series :

Format

Label	Command	Argument
[<Label >]	LDC	= CTX. T : TST
	STR	= REG. X
	CHX	
	LDC	= TST . < item name > , X
	or	
	STC	= TST . < item name > , X

Generated Code

```
[<Label >]  LDC     = <associated value 1>
            STR     = <associated value 2>
            CHX
            LDC     = <associated value 3>  , X
            or
            STC     = <associated value 3>  , X
```

User Interface

Addressable items are :

B:PGS Program Segment Base

L:PGS Program Segment Length

B:SLS	Program Overlay Management Table Base
L:SLS	Program Overlay Management Table Length
B:Z	Data Segment Base At Time Of Call To Supervisor
L:Z	Data Segment Length At Time Of Call To Supervisor
B:SPS	Subroutine Segment Base
L:SPS	Subroutine Segment Length
B:BS1	Base Of Non-Specialized Segment 1
L:BS1	Length Of Non-Specialized Segment 1
B:BS2	Base Of Non-Specialized Segment 2
L:BS2	Length Of Non-Specialized Segment 2

IV-4. ACCESS TO HARDWARE ENVIRONMENT STRUCTURES

IV-4.1. Access To The General Registers Structure

General registers are described by a simple structure called REG.

When a general register is to be accessed, one of the following two internal macros is used :

Format

Label	Command	Argument
[< Label >]	LDR	= REG . < register name >
	or	
[< Label >]	STR	= REG . < register name >

Generated Code

```
[< Label >]  LDR     = < associated value >
             or
             STR     = < associated value >
```

User Interface

Addressable general registers are :

P	: Program Counter
L	: Current Address Of Local Data Area
A	: Accumulator
E	: Accumulator Extension
X	: Index Register
ITL	: Interrupt Management Register

IV-4.2. Access To The Base Registers Structure

Base registers are described by a simple structure called BASE.

When a base register is to be accessed, one of the following two internal macros is used :

Format

Label	Command	Argument
[< Label >]	LDB	= BASE . < register name >
	or	
[< Label >]	STB	= BASE . < register name >

Generated Code

```
[< Label >]  LDB     = < associated value >
             or
[< Label >]  STB     = < associated value >
```

User Interface

Addressable base registers are :

G	: Program Segment Base
GL	: Program Segment Length
Z	: Data Segment Base
ZL	: Data Segment Length
C	: Context Segment Base
Q	: Subroutine Segment Base
QL	: Subroutine Segment Length

IV-5. ACCESS TO A STRUCTURE ORGANIZED AS A SEGMENT AND LOCATED IN A SEGMENT THAT IS NOT DIRECTLY ADDRESSABLE

To access a segment that is not part of the directly addressable segments, its descriptor must be loaded in an appropriate register pair.

A base/length register pair may be loaded :

- . using instruction LDG for pair (G, GL)
- . using instruction LDZ for pair (Z, ZL)

The addressing mode that permits access to the descriptor is special for these two instructions. It distinguished two types of descriptor tables :

- . table TST which is defined at the level of each task in the context segment
- . table GST which is defined at the group level

Instructions LDG and LDZ designate the descriptor to be loaded by its number in one of these two tables.

IV-5.1. Access To A Structure Organized As A Segment Based By G

IV-5.1.1. Access To A Structure

If the structure belongs to a segment based by G, it may be based by G using one of the following 2 access macros :

- . ACCESS, G < structure name >
- . ACCESS, G < structure name > , X

Common Interface

User must reserve in CDS or in LDS of the section accessing the structure a constant containing the segment configuration :

K : < configuration > DATA & < configuration >

After the access, base G points to the beginning of the segment.

Format 1 Of Access Macro

Label	Command	Argument
	ACCESS, G	< structure name >

Generated Code

```
LDA     K :     < configuration >
LDG
```

Input Interface

User shall reserve a constant containing the segment configuration :

K : < configuration > DATA & < configuration >

Segment configuration is as follows :

Bit 0 = 0, for a direct segment, i.e., a segment whose descriptor is in the TST of the context.

= 1, for an indirect segment, i.e., whose descriptor is in the GST of the context.

Bits 1 - 7 = 0

Bits 8 - 15 = number of segment descriptor in the TST or GST of the context.

Output Interface

Base G points to the beginning of the segment.

Format 2 Of Access Macro

Label	Command	Argument
	ACCESS, G	< structure name > , X

Generated Code

```
LDA     K :  < configuration >
LDG
```

Input Interface

Register X contains the number of the structure addressed within the segment.

User shall reserve a constant containing the segment configuration.

K : < configuration > DATA & < configuration >

Segment configuration is as follows :

Bit 0 = 0, for a direct segment, i.e., whose descriptor is in the TST of the context.

= 1, for an indirect segment, i.e., whose descriptor is in the GST of the context.

Bits 1 - 4 = 0

Bits 5 - 7 = exponent n of length 2^n (in bytes) of the structure.

Bits 8 - 15 = number of segment descriptor in the TST or GST of the context.

Output Interface

Base G points to the beginning of the segment.

Register X contains the displacement of the structure addressed within the segment.

Structure length (in bytes) must be a power of two.

IV-5.1.2. Access To An Item Of The Structure

An item of the structure is accessed using internal macros with the following format :

Label	Command	Argument
[< Label >]	< Operation Code >	@#< structure name > . < item name > , X

Generated Code

[< Label >] < Operation Code > @#< value > , X

User Interface

Register X contains the displacement of the structure addressed within the segment to which it belongs.

< value > is the address of the pointer of the item of the 1st segment structure.

< value > = BPA + 2 X (item rank)

Topology of access to a segment structure based by G is given in Figure 4-2.

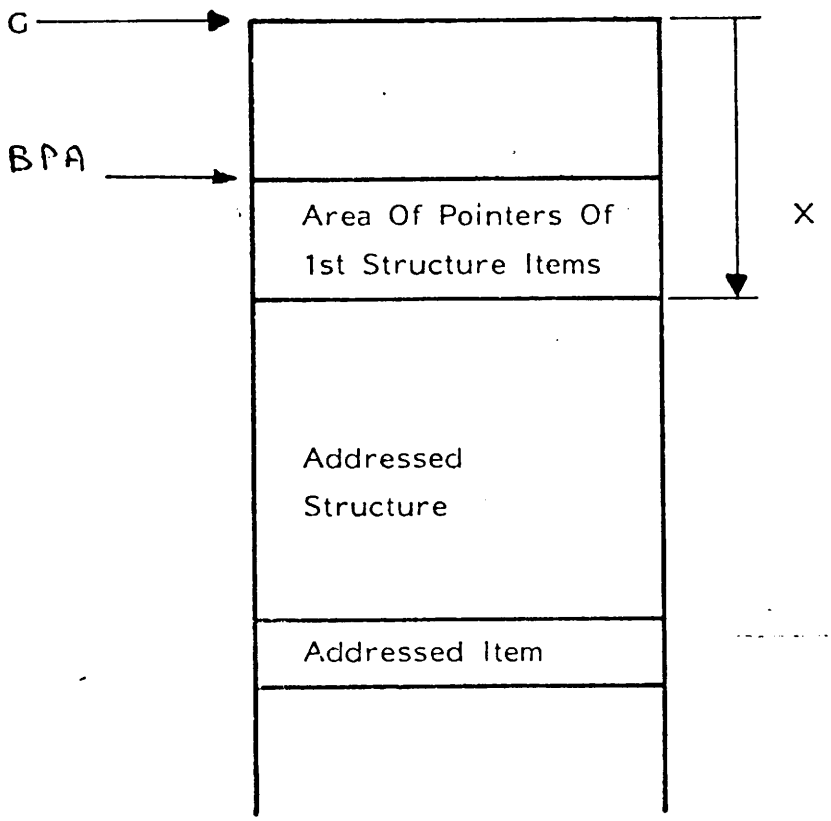


Figure 4 - 2. Topology Of Access To A Segment Structure Based By G.

IV-5.2. Access To A Structure Organized As A Segment Based By Z

IV-5.2.1. Access To Structure

If the structure belongs to a segment based by Z, it may be based by Z using one of the following two access macros :

. ACCESS, Z < structure name >
. ACCESS, Z < structure name >, X

Common Interface

User must reserve in the CDS or LDS of the section accessing the structure a constant containing the segment configuration :

K : < configuration > DATA & < configuration >

After access, base Z points to the beginning of the segment.

Format 1 Of Access Macro

Label	Command	Argument
	ACCESS, Z	< structure name >

Generated Code

LDA K : < configuration >
LDZ

Input Interface

User shall reserve a constant containing the segment configuration :

K : < configuration > DATA & < configuration >

Segment configuration is as follows :

Bit 0 = 0, for a direct segment, i.e., whose descriptor is in the TST of the context.

= 1, for an indirect segment, i.e., whose descriptor is in the GST of the context .

Bits 1 - 7 = 0

Bits 8 - 15 = number of segment descriptor in the TST or GST of the context.

Output Interface

Base Z points to the beginning of the segment.

Format 2 Of The Access Macro

Label	Command	Argument
	ACCESS, Z	< structure name> , X

Generated Code

```
LDA     K : < configuration >
LDZ
```

Input Interface

Register X contains the number of the structure addressed within the segment.

User shall reserve a constant containing the segment configuration :

K : < configuration > DATA & < configuration >

Segment configuration is as follows :

Bit 0 = 0, for a direct segment, i.e., whose descriptor is in the TST of the context.

= 1, for an indirect segment, i.e., whose descriptor is in the GST of the context.

Bits 1 - 4 = 0

Bits 5 - 7 = exponent n of length 2^n (in bytes) of the structure.

Bits 8 - 15 = number of segment descriptor in the TST or GST of the context.

Output Interface

Base Z points to the beginning of the segment.

Register X contains the displacement of the structure addressed within the segment.

Structure length (in bytes) must be a power of two.

IV-5.2.2. Access To A Structure Item

An item of a structure is accessed using internal macros with the following format :

Label	Command	Argument
[< Label >]	<Operation Code >	$< structure name > . < item name >[, X]

Generated Code

If operation code is an instruction addressing the word :

[< Label >] < operation code > $ K : < associated value + 1 >[, X]

If< operation code >is an instruction addressing the byte :

[< Label >] < operation code > $Z : < associated value > [, X]

User Interface

A structure can be accessed from the program segment, subroutine segment, or supervisor segment.

X must contain - prior to access - the byte displacement of the addressed structure relatively to the beginning of the segment based by Z. If this displacement is zero (structure at beginning of segment), X may be omitted in the "item access" macro ; it will not appear in the generated instruction.

User must generate in the CDS or the LDS of the section accessing the structure :

. logical word pointers relative to Z whose displacements are values associated with word items of the structure.

K : < associated value + 1 > DATA & < associated value + 1 >

. logical byte pointers relative to Z whose displacements are values associated

with the bytes items of the structure.

Z : < associated value > PTB $ & < associated value >

Topology of access to a segment structure based by Z is given in Figure 4-3.

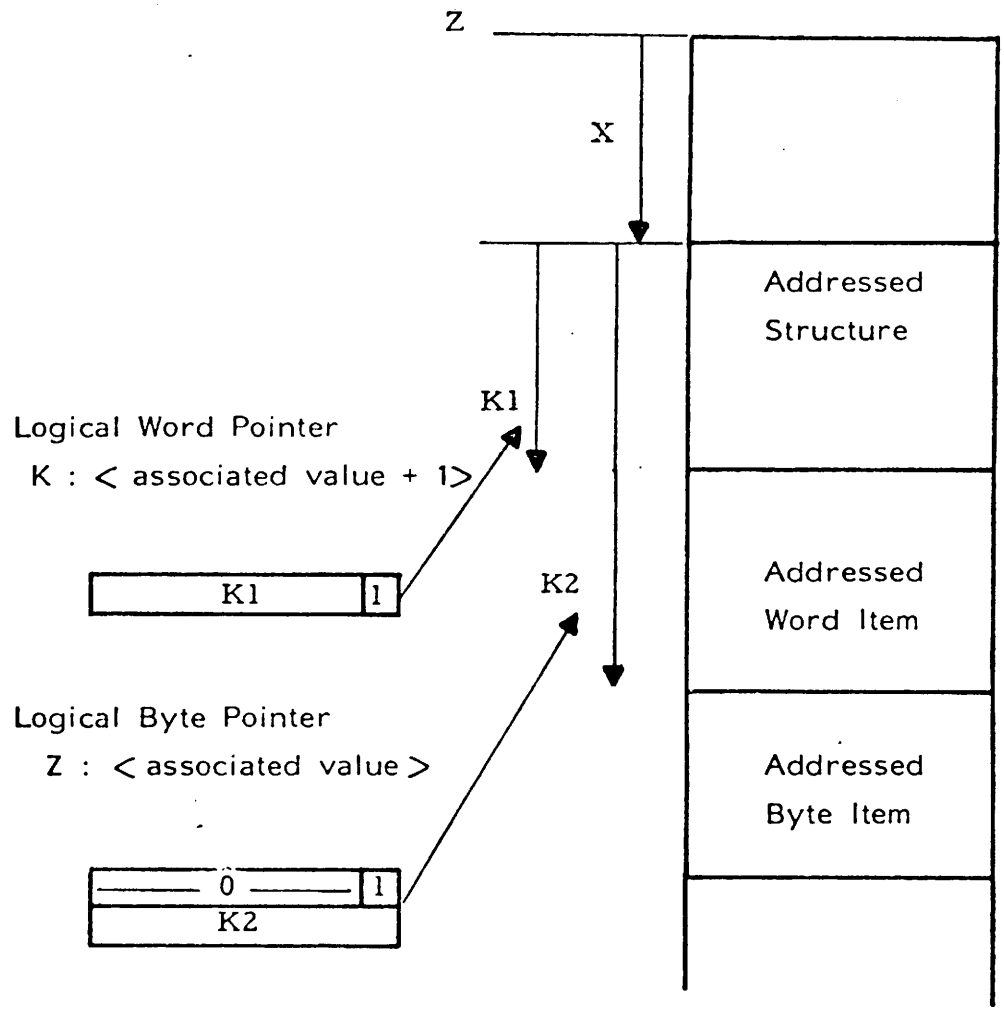


Fig. 4-3 Topology of access to A segment structure based by Z

V - INSTRUCTION DESCRIPTION

V-1. GENERAL

This chapter describes MITRA 125 Privileged mode instructions (PV = 1) in alphabetical order.

The following are indicated for each instruction :

- instruction name
- instruction class (the instruction class indicates the type of addressing permitted)
- instruction code as a function of the type of addressing used
- instruction function
- elements modified by the instruction (memory, registers, indicators)
- possible traps

MITRA 125 "non-privileged" instructions usable in Privileged mode are described in Volume I of this manual.

V-2. PRIVILEGED INSTRUCTION DESCRIPTION

V-2.1. Symbolic Notations Used

A	: Accumulator register or its contents
$\overline{A}$	: Logical complement of the contents of A
A_{0-7}	: Most significant (left) byte of A
A_{8-15}	: Least significant (right) byte of A
E	: Extension register or its contents
E,A	: Extended register made up of Accumulator A and Extension register E, with the most significant bits in E, or their contents
X	: Index register
R_n	: Register n or its contents
L	: Local Base register or its contents
P	: Program counter or its contents
C	: Instruction result indicator (carry)
O	: Instruction result indicator (overflow)

SP : Subroutine mode indicator

PV : Privileged mode indicator

SV : Supervisor mode indicator

PR : Memory protection indicator

MA : Interrupt mask indicator

G : Program segment base register

GL : Program segment length register

Q : Subroutine segment base register

QL : Subroutine segment length register

Z : Data segment base register

ZL : Data segment length register

G' : G in Program mode
Q in Subroutine mode
S in Supervisor mode

B : G or Z in extended addressing depending on the base referenced by the logical pointer

D : Displacement (least significant byte of the instruction extended to one word per a zero most significant byte)

IN : Instruction

(I) : Contents of I

I/J : Address I relative to base J

(I)/J : Contents of memory cell with address I/J

N : Operand-computed value

Y : Computed address relative to the beginning of the referenced segment (Y = f (D) with f = addressing function)

y_1 : Byte of address Y/G or Y/Q or Y/S or Y/Z depending on the operating mode and the type of addressing

y_2 : Address word :
. if Y even : Y/G, Y/Q, Y/S or Y/Z
. if Y odd : Y-1/G, Y-1/Q, Y-1/S or Y-1/Z

Y_1 : absolute address of memory cell containing byte y_1 : $(Y_1) = y_1$

Y_2 : Absolute address of memory cell containing word y_2 : $(Y_2) = y_2$

$I \rightarrow J$: J assumes value I

$I \leftrightarrow J$: J assumes value I and I assumes value J

+ : Exclusive OR operation

$\wedge$: Logical AND operation

v : Inclusive OR operation

$\oplus$: Plus

- : Minus

*	: Multiply
/	: Divide

V-2.2. Privileged Instruction List

AIO	: Acknowledge I/O interrupt
BRK	: Generate illegal instruction trap (Break)
CLM	: Clear mask
CLP	: Clear memory protection
DELQ	: Delete an element in queue
DIT	: Deactivate interrupt
EXEC	: Execute a designated instruction
HIO	: Halt I/O
INQ	: In queue
INQP	: In queue with priority
LDB	: Load Base
LDC	: Load Context element
LDG	: Load G and GL
LDZ	: Load Z and ZL
OUTQ	: Queue-out first queue element
RD	: Read Direct
RSV	: Return from Supervisor
RTD	: Return from trap
SIO	: Start I/O
STB	: Store in a Base
STC	: Store in a Context element
STI	: Store in the Indicators
STM	: Mask interrupt
STP	: Set memory Protection
STR	: Store Register
TESTQ	: Test first Queue element
WD	: Write Direct
XCTX	: Exchange context

Name : Acknowledge Input/Output interrupt

Class : 1' Family SYS Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / F406

Function

Instruction AIO permits identifying the device that has caused the interrupt and the interrupt causes.

Register E	Bit 0	Value 0
	Bits 1-7	Controller number
	Bits 8-15	Number of I/O message words
Register A	Bits 0-15	Address of I/O message relative to base located in register X
Register X	Bits 0-15	Base of I/O message address

An input/output message is returned if AIO is accepted and the interrupt level is not deactivated.

Modified Elements

Indicators C and O

Indicators C And O After Execution

C = 0 and O = Don't Care	: AIO accepted	: normal interrupt
C = 1 and O = 0	: AIO accepted	: normal end
C = 1 and O = 1	: AIO rejected	: interrupt level other than ADM or parasitic

Trap

An illegal instruction trap occurs if bit zero of register E is other than zero.

A mode violation trap occurs if the program executing AIO is not in Privileged mode.

Name : BreaK

Class : 1' Family COV Privileged Mode

Instruction Code

Code : Bits 0 - 11
Displacement : Bits 12 - 15

Addressing Mode/Hexadecimal Code : P / FF40

Function

This instruction generates an illegal instruction trap.

Registers P and L and indicators are saved in words 4, 6, and 8 of the supervisor segment.

Word of address 4/S gives the address of instruction BRK.

Remark

Bits 12-15 of the computed operand are not decoded by hardware and may be used by software.

Trap

An illegal instruction trap occurs when instruction BRK is executed.

Name : CLear Mask

Class : 1' Family SYS Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / F400

Function

This instruction clears all interrupts ; value zero is stored in indicator MA.

Modified Elements

Indicator MA

Trap

A mode violation trap occurs if the program executing CLM is not in Privileged mode.

Name : CLear Memory Protection

Class : 1' Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / F407

Function

Instruction CLP sets to zero the protection bit of a word string in memory.

Address relative to Z of the first word of this string is in register A.

Length of this string expressed in number of words is in register E.

At end of execution : E = - 1
A = Address of the 1st word not processed

Modified Elements

Registers E and A
Memory locations Y_2 through $(Y_2 + E - 1)$

Traps

A mode violation trap occurs if the program executing CLP is not in Privileged mode.

Remark

This instruction is interruptible between each word protection removal.

Name : DELete an element in Queue

Class : 1' Family COV Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / FF23

Function

Extract (queue-out) an element QUE (Queue Element) from the queue to which it belongs.

Element QUE comprises 5 control words followed by a message of any length.

Register E contains the address of the QUE element relative to the head of segment QUS (Queue Segment).

The descriptor of segment QUS is in the first two words of segment HTS (Head Tail Segment) described by the pair (Base Z and Length ZL) ; the first word contains the base of segment QUS ; the second word contains the length of segment QUS.

Modified Elements

The first control byte (CODE) of element QUE is not modified ; it is copied in the leftmost byte of register A (Bits 0 - 7).

The second control byte (PRTY) of element QUE is not modified ; it is copied in the rightmost byte of register A (Bits 8 - 15).

Second, third, fourth, and fifth control words of element QUE are set to zero.

The leftmost byte of register X (Bits 0 - 7) is set to zero.

The rightmost byte of register X (Bits 8 - 15) is loaded with the second control byte (PRTY) of element QUE.

Base register Z is loaded with the base of segment HTS which contained double-word HTE (Head Tail Element) associated with the queue in which the element QUE was located.

Length register ZL is loaded with &FFFF.

Chaining words of elements belonging to the same queue are updated.

Double-word HTE is updated if element QUE was the first or last element of its queue.

Indicators C And O After Execution

C = 0 and O = 0	: Queue is not empty
C = 0 and O = 1	: Queue becomes empty
C = 1 and O = Don't Care	: Element does not belong to any queue

Traps

A mode violation trap occurs if the program executing DELQ is not in Privileged mode.

A length overflow trap occurs if $E_{0-15} + 10 > L$ (segment QUS)

Name : Deactivate InterrupT

Class : 1' Family SYS Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / F401

Function

Instruction DIT deactivates the current interrupt and triggers acceptance of the highest-priority waiting interrupt.

Value of register P is incremented by 2.

Current interrupt level is deactivated.

Indicators and registers X, E, A, L, P, G, GL, Z, and ZL are stored in the first 10 words of the context segment if bit 15 of word ACPT is set (to 1).

A new context segment is used by loading base register C with the base of the context associated with the highest-priority waiting interrupt (this base is in table CPT located in the supervisor segment).

The first 12 words of the new context segment are stored in the indicators and registers X, E, A, L, P, G, GL, Z, ZL, Q, and QL.

Rightmost byte of register RIT is loaded with rank and level of accepted interrupt ; leftmost byte of register RIT is loaded with rank and level of the interrupt that was being processed before the current interrupt.

Modified Elements

Indicators
Registers X, E, A, L, P, G, GL, Z, ZL, Q, QL
Register RIT
Base C

Traps

A mode violation trap occurs if the program executing instruction DIT is not in Privileged mode.

Miscellaneous

Any interrupt program is terminated by a DIT.

A DIT has no meaning at level 0.

There are 128 interrupts divided into 32 levels of 4 ranks each.

The effect of instruction DIT does not depend on the state of flip-flop MA or the indicator light RUN.

Name : EXECute a designated instruction

Class : 1' Family COV Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / FF26

Function

This instruction unlocks traps.

After registers P, L and the indicators are loaded with the contents of the words located at addresses &1E, &20, and &22 of the context segment based by C, the instruction designated by the new value of the program counter P and the indicators is executed.

After the instruction is executed, an "end of instruction EXEC" trap is generated.

Traps

A mode violation trap occurs if the program executing instruction EXEC is not in Privileged mode.

Name : Halt Input/Output

Class : 1' Family SYS Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / F 405

Function

This instruction orders the end of transfer on the peripheral addressed by register E.

Register E	Bit 0	Value 0
	Bits 1 - 7	Controller number
	Bits 8 - 15	Device number

Modified Elements

Indicators C and O

Indicators C And O After Execution

C = 0 and O = Don't Care	: HIO accepted	: Device busy
C = 1 and O = 0	: HIO transparent	: Device idle
C = 1 and O = 1	: HIO rejected	: Non-existent Device

Traps

An "illegal instruction" trap occurs if bit zero of register E is other than zero.

A mode violation trap occurs if the program executing HIO is not in Privileged mode.

Remark

HIO is accepted only if the addressed device is busy.

An interrupt will be generated as soon as the operation on the device is terminated ; this end may be a normal channel program end or an end caused by HIO.

The distinction is made with instruction AIO executed under an interrupt.

Name : IN Queue

Class : 1' Family COV Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / FF20

Function

Insert element QUE (Queue Element) in the queue defined by HTE (Head Tail Element) according to the priority defined by byte PRTY of element QUE.

Element QUE comprises 5 control words followed by a message of any length.

Element QUE is defined by its address relative to the head of segment QUS (Queue Segment).

Register E contains the relative address of element QUE.

Descriptor of segment QUS is in the first two words of segment HTS (Head Tail Segment) described by pair (Z, ZL) ; the first word contains the base of segment QUS ; the second word contains the length of segment QUS.

Register X contains the relative address of element HTE in segment HTS.

The queue is analyzed starting from the end.

Remark

Register E must be other than zero.

Modified Elements

Element QUE is inserted between the last element with priority greater than or equal to that of QUE and the element with priority less than QUE.

The first control byte (CODE) of the element is loaded with the leftmost byte of register A (A_{0-7}).

The second control byte (PRTY) of the element is not modified ; it is copied in the rightmost byte of register A (A_{8-15}) ; after execution, the first control word of element QUE and register A are therefore identical ; the second and third control words of the element are modified according to chaining in the queue.

Register Z (base of segment HTS) is copied in the fourth control word of the element.

Register X (address relative to segment HTS of HTE) is copied in the fifth control word of the element.

Double-word HTE of segment HTS is updated if element QUE is the head or tail element of the queue.

Indicators C And O After Execution

C = 0 and O = Don't Care : Queue-in correct

C = 1 and O = Don't Care : Element already queued

Traps

A mode violation trap occurs if the program executing INQ is not in Privileged mode.

A length overflow trap occurs if $X_{0-15} + 4 > ZL$ or if $E_{0-15} + 10 > L$ (Segment QUS).

Name : IN Queue with Priority

Class : 1' Family COV Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / FF24

Function

Insert element QUE (Queue Element) in queue defined by HTE (Head Tail Element) according to a priority defined in register A.

Element QUE is divided into 5 control words followed by a message of any length.

Register E contains the address of element QUE relative to the head of segment QUS (Queue Segment).

The descriptor of segment QUS is in the first two words of segment HTS (Head Tail segment) described by pair (Z, ZL) ; the first word contains the base of segment QUS ; the second word contains the length of segment QUS.

Register X contains the relative address of element HTE in segment HTS.

The queue is analyzed starting from the end.

Remark

Register E must be other than zero.

Modified Elements

Element QUE is inserted between the last element with priority greater than or equal to QUE and the element with priority less than QUE.

First control byte (CODE) of the element is loaded with the leftmost byte of register A (A_{0-7}).

Second control byte (PRTY) of the element is loaded with the rightmost byte of register A (A_{8-15}).

Second and third control words of the element are modified according to the chaining in the queue.

Register Z (base of segment HTS) is copied in the fourth control word of the element.

Register X (address relative to segment HTS of HTE) is copied in the fifth control word of the element.

Double-word HTE of segment HTS is updated if element QUE is the head or tail element of the queue.

Indicators C And O After Execution

C = 0 and O = Don't Care : Queue-in correct

C = 1 and O = Don't Care : Element already queued

Traps

A mode violation trap occurs if the program executing INQP is not in Privileged mode.

A length overflow trap occurs if $X_{0-15} + 4 > ZL$ or if $E_{0-15} + 10 > L$ (Segment QUS).

Name : LoaD Base

Class : 1' Family MCB Privileged Mode

Instruction Code

Code : Bits 0 - 9
Displacement : Bits 10 - 15

Addressing Mode/Hexadecimal Code :

Parameter / FB00
Indexed Parameter / FB40

Function

This instruction loads register A with the contents of base register n if n is other than 62 and with zero if n equals 62 (base register S).

In parameterized mode : $n = IN_{10-15}$

In indexed parameterized mode : $n = IN_{10-15} + X_{0-15}$

The value of n must be between 0 and 63 ; addressing of an address base greater than 63 yields an undetermined result.

Modified Elements

Register A

Traps

A mode violation trap occurs if the program executing LDB is not in Privileged mode.

Remark

Base registers all have 16 bits except for registers 33 through 63 which have only bit 15.

Name : LoaD Context

Class : 1' Family MCB Privileged Mode

Instruction Code

Code : Bits 0 - 9
Displacement : Bits 10 - 15

Addressing Mode/Hexadecimal Code

Parameter / EB00
Indexed Parameter / EB40

Function

This instruction loads register A with the contents of a context element whose address relative to base C of the context is supplied by the displacement field.

$Y/C \rightarrow A$

In parameterized mode : $Y/C = 2 \times (IN_{10-15})$

In indexed parameterized mode : $Y/C = 2 \times (IN_{10-15} + X_{0-15})$

Modified Elements

Register A
Indicators C and O

Indicators C And O After Execution

C = 0 and O = 0	A > 0
C = 0 and O = 1	A < 0
C = 1 and O = 0	A = 0

Trap

A mode violation trap occurs if the program executing LDC is not in privileged mode.

Remark

In parameterized mode, it is therefore possible to address 64 words.
In indexed parameterized mode, register X contains a word displacement.

Name : LoaD Descriptor (G, GL)

Class : 1' Family COV Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / FF10

Function

Load base G and length GL using a descriptor from table GST (Table of descriptors of segments common to a task group) or TST (Table of descriptors of segments specific for the task).

If $A_0 = 0$, A_{8-15} gives the number of the segment descriptor in TST.
If $A_0 = 1$, A_{8-15} gives the number of the segment descriptor in GST.

Logical shift left n bit positions in register X with $n = A_{5-7}$. If during the shift, one bit set at 1 is lost, the contents of register X becomes non-significant.

Modified Elements

Base G and GL registers
Register X
Indicators C and O

Indicators C And O After Execution

C = 0 and O = Don't Care : Load correct
C = 1 and O = 0 : GL = 0
C = 1 and O = 1 : GL ≠ 0 and bits set to 1 lost during shift of register X
GL ≠ 0 and X shifted ≥ GL

Remark

No check performed if register X is not shifted ($n = A_{5-7} = 0$)

Traps

A mode violation trap occurs if the program executing instruction LDG is not in Privileged mode.

Name : LoaD Descriptor (Z, ZL)

Class : 1' Family COV Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / FF10

Function

Load base Z and length ZL using a descriptor from table GST (Table of descriptors of segments common to a task group) or TST (Table of descriptors of segments specific for the task).

If $A_0 = 0$, A_{8-15} gives the number of the segment descriptor in TST.
If $A_0 = 1$, A_{8-15} gives the number of the segment descriptor in GST.

Logical shift left n bit positions in register X with $n = A_{5-7}$. If during the shift, one bit set at 1 is lost, the contents of register X becomes non-significant.

Modified Elements

Base Z and ZL registers
Register X
Indicators C and O

Indicators C And O After Execution

C = 0 and O = Don't Care : Load correct
C = 1 and O = 0 : ZL = 0
C = 1 and O = 1 : ZL ≠ 0 and bits set to 1 lost during shift of register X
ZL ≠ 0 and X shifted ≥ ZL

Remark

No check performed if register X is not shifted ($n = A_{5-7} = 0$)

Traps

A mode violation trap occurs if the program executing instruction LDZ is not in Privileged mode.

Name : OUT Queue

Class : 1' Family COV Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / FF21

Function

Extract (queue-out) first QUE element (Queue Element) of queue defined by HTE (Head Tail Element).

Element QUE comprises 5 control words followed by a message of any length.

Register X contains the relative address of element HTE in segment HTS (Head Tail Segment).

Base Z and length ZL registers contain the base and length of segment HTS.

Modified Elements

First control byte (CODE) of element QUE is not modified ; it is copied in the leftmost byte of register A (A_{0-7}).

Second control byte (PRTY) of element QUE is not modified ; it is copied in the rightmost byte of register A (A_{8-15}).

Second, third, fourth, and fifth control words of element QUE are set to zero.

Register E gives the relative address of element QUE in segment QUS.

The leftmost byte of register X (X_{0-7}) is set to zero.

The rightmost byte of register X (X_{8-15}) is loaded with the second control byte (PRTY) of element QUE.

Double-word HTE of segment HTS is updated.

Indicators C And O After Execution

C = 0 and O = 0 : Queue is not empty
C = 0 and O = 1 : Queue becomes empty
C = 1 and O = Don't Care : Queue was empty

Traps

A mode violation trap occurs if the program executing OUTQ is not in Privileged mode.

A length overflow trap occurs if $X_{0-15} + 4 > ZL$.

Name : Read Direct

Class : 1' Family SYS Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / F402

Function

This instruction reads the status word or data from a coupler.

Register E : Coupler address and function to be performed

Register A : Data read and status returned

In general, this instruction is to be executed during an interrupt generated after an order.

Traps

A mode violation trap occurs if the program executing RD is not in Privileged mode.

Name : Return from Supervisor

Class : 1' Family SRG Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / F10C

Function

This instruction is executed in a supervisor section called by a CSV. It restores indicators saved in the third word of the CDS at the time of CSV and loads values Pi + 2 and Li (Pi and Li being the contents of the first two words of the CDS) in registers P and L.

(G + 4) → Indicators
G + (G + 2) → L
G + 2 + (G) → P

Modified Elements

Registers L and P
Indicators

Traps

A mode violation trap occurs if the program executing RSV is not in Privileged mode.

Name : Return from Trap

Class : 1' Family COV Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / FF25

Function

This instruction unlocks traps.

Registers P, L and indicators are loaded with the contents of words located at addresses &1E, &20, and &22 of the context segment based by C.

Traps

A mode violation trap occurs if the program executing RTD is not in Privileged mode.

Name : Start Input/Output

Class : 1' Family SYS Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / F404

Function

This instruction starts a channel program on the peripheral addressed by registers E, A, and X.

REGISTER E	Bit 0	Value 0
	Bits 1 - 7	Controller number
	Bits 8 - 15	Device number
REGISTER A	Bits 0 - 15	Channel program address relative to the base located in register X
REGISTER X	Bits 0 - 15	Base of channel program address

Modified Elements

Indicators C and O

Indicators C And O After Execution

C = 0 and O = Don't Care : SIO accepted
C = 1 and O = 0 : SIO rejected : Device busy
C = 1 and O = 1 : SIO rejected : Non-existent device

Remark

SIO is accepted only if the device addressed is idle ; in this case, the device goes to the busy status.

An interrupt will be generated each time it is requested in the channel command and at the end of the channel program.

Traps

An illegal instruction trap occurs if bit zero of register E is other than zero. A mode violation trap occurs if the program executing SIO is not in Privileged mode.

Name : STore A in a Base

Class : 1' Family MCB Privileged Mode

Instruction Code

Code : Bits 0 - 9
Displacement : Bits 10 - 15

Addressing Mode/Hexadecimal Code

Parameter / FB80
Indexed Parameter / FBC0

Function

This instruction stores the value of register A in base register Bn if $n \neq 62$; it stores A + S in base register S if n = 62.

In parameter mode : $n = IN_{10-15}$

In indexed parameter mode : $n = IN_{10-15} + X_{0-15}$

Only bases 0 through 63 exist ; addressing an address base greater than 63 yields an undetermined result.

Modified Elements

Base Bn if $n \neq 62$
Base register S if n = 62

Traps

A mode violation trap occurs if the program executing STB is not in Privileged mode.

Name : STore A in a Context element

Class : 1' Family MCB Privileged Mode

Instruction Code

Code : Bits 0 - 9
Displacement : Bits 10 - 15

Addressing Mode/Hexadecimal Code

Parameter / EB80
Indexed Parameter / EBC0

Function

This instruction stores register A in a context element pointed relatively to base C of the context :

$A \rightarrow Y/C$

In parameterized mode : $Y/C = 2 \times (IN_{10-15})$
In indexed parameterized mode : $Y/C = 2 \times (IN_{10-15} + X_{0-15})$

Modified Elements

Context element

Traps

A mode violation trap occurs if the program executing STC is not in Privileged mode.

Remark

In parameterized mode, it is therefore possible to address 64 words of the context.

In indexed parameterized mode, register X must contain a word displacement.

Name : STore register A in Indicators

Class : 1' Family SRG

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / F132

Function

This instruction loads the indicators with the contents of register A.

In Privileged mode, all indicators can be loaded.

A_{15} → O
A_{14} → C
A_{13} → PV
A_{12} → MA
A_{11} → PR
A_{9} → SP
A_{8} → SV

Modified Elements

Indicators

Name : SeT interrupt Mask

Class : 1' Family SYS Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / F408

Function

This instruction masks all interrupts. Value "one" is stored in indicator MA.

1 → MA

Modified Elements

Indicator MA

Traps

A mode violation trap occurs if the program executing STM is not in Privileged mode.

Name : STore memory Protection

Class : 1' Family SYS Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / F40F

Function

This instruction sets the protection bit of a word string in memory to one.

Address relative to Z of the first word of this string is in register A.

Length of this string expressed in number of words is in register E.

After execution :

E = - 1

A = Address of 1st unprocessed word.

Modified Elements

Registers E and A

Memory locations : Y_2 through (Y_2 + E - 1)

Traps

A mode violation trap occurs if the program executing STP is not in Privileged mode.

Remark

This instruction is interruptible between each word protection.

Name : STore A in a Register

Class : 1' Privileged Mode

Instruction Code

Code : Bits 0 - 7
Displacement : Bits 8 - 15

Addressing Mode/Hexadecimal Code

DL	3A00
PX	EA00
P	FA00

Function

This instruction stores register A in register Rn with n = (Y).

A → Rn

Modified Element

Register Rn

Traps

A mode violation trap occurs if the program executing STR is not in Privileged mode.

Name : TESt first element of a Queue

Class : 1' Family COV Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / FF22

Function

Test the first QUE element (Queue Element) of queue defined by HTE (Head Tail Element).

Register X contains the relative address of element HTE in segment HTS (Head Tail Segment).

Base Z and length ZL registers contain the base and length of segment HTS.

Modified Elements

The leftmost byte of register A (A_{0-7}) gives the first control byte (CODE) of element QUE.

The rightmost byte of register A (A_{8-15}) gives the second control byte (PRTY) of element QUE.

Register E gives the relative address of element QUE in segment QUS.

The leftmost byte of register X (X_{0-7}) is set to zero.

The rightmost byte of register X (X_{8-15}) gives the second control byte (PRTY) of element QUE.

Indicators C And O After Execution

C = 0 and O = 0	: Element QUE is not alone in the queue
C = 0 and O = 1	: Element QUE is the only element in the queue
C = 1 and O = Don't Care	: Queue is empty

Traps

A mode violation trap occurs if the program executing TESQ is not in Privileged mode.

A length overflow trap occurs if $X_{0-15} + 2 > ZL$.

Name : Write Direct

Class : 1' Family SYS Privileged Mode

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / F403

Function

This instruction transmits an order or data to a coupler.

Register E : coupler address and order to be executed
Register A : data and complementary order

In general, each order to a coupler causes an interrupt and an order executed under an interrupt is accepted only when the interrupt is deactivated (DIT).

Traps

A mode violation trap occurs if the program executing WD is not in Privileged mode.

Name : eXchange ConTeXt

Class : 1' Privileged Mode Not Subroutine

Instruction Code

Code : Bits 0 - 15

Addressing Mode/Hexadecimal Code : P / FE00

Function

Store indicators and registers X, E, A, L, P, G, GL, Z, ZL in the first 10 words of the context segment.

Store register A in word i of the CPT (table containing the 128 bases of the 128 context segments associated with the 128 interrupts) - 2i designating the contents of register RIT (Interrupt Register).

Load base C with the contents of register A.

Call the first 12 words of the new context segment to load indicators and registers X, E, A, L, P, G, GL, Z, ZL, Q, QL.

Traps

A mode violation trap occurs if the program executing XCTX is not in Privileged mode.

Remark

The instruction with the same code as XCTX in Privileged mide is instruction CLQ in shared program mode (with $N_{8-15} \neq 0$).

cii COMPAGNIE INTERNATIONALE
POUR L'INFORMATIQUE

DIVISION MILITAIRE SPATIALE ET AERONAUTIQUE

10 - 12 Avenue de l'Europe - 78140 VELIZY (France)
Tél. *946.96.70*