APPENDIX A LIMITATIONS

A.1 Limitation on Conflict between Macro Service and INT

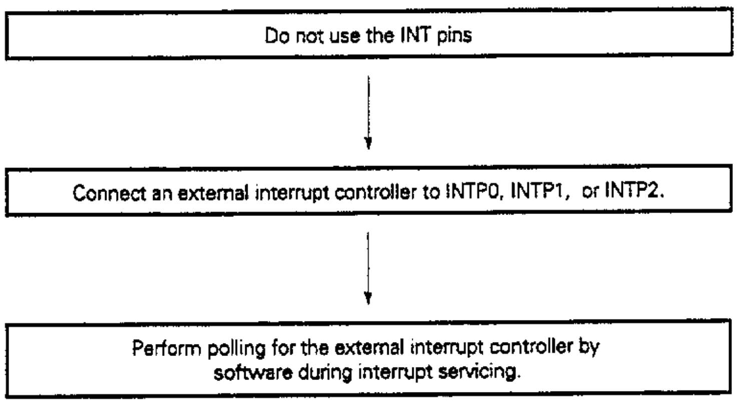
A.1.1 Description of limitation

If an INT request occurs while a macro service request is held pending under interrupt priority control, the macro service which should be pending may instead be executed first.

A.1.2 Avoidance methods

If this symptom adversely affects your system when using the macro service function, do not use the macro service function and INT input at the same time.

If this symptom adversely affects your system when an external interrupt controller (μPD71059) is connected and the macro service function is used, perform interrupt servicing as shown below.

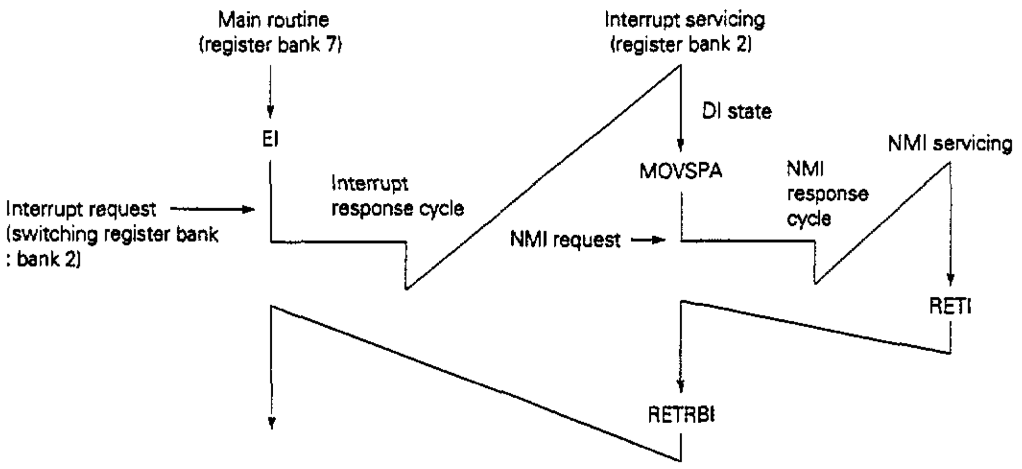

Do not use the INT pins
Connect an external interrupt controller to INTP0, INTP1, or INTP2.
Perform polling for the external interrupt controller by software during interrupt servicing.

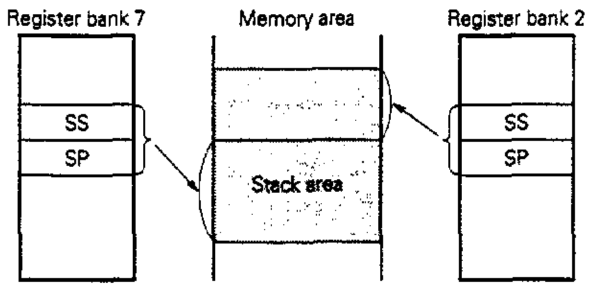
A.2 Cautions on Use of the MOVSPA Instruction

A.2.1 Device operation description (symptom)

In applications where register banks before and after switching use contiguous stack areas when register bank switching is performed in an asynchronously occurring interrupt, the MOVSPA instruction can be used to copy SS and SP between the register banks before and after switching. In this case, the stack areas used before and after register bank switching can be made contiguous by executing the MOVSPA instruction at the beginning of program processing of the register bank after switching. (See **Figure A-1.**)

Figure A-1. Normal Operation

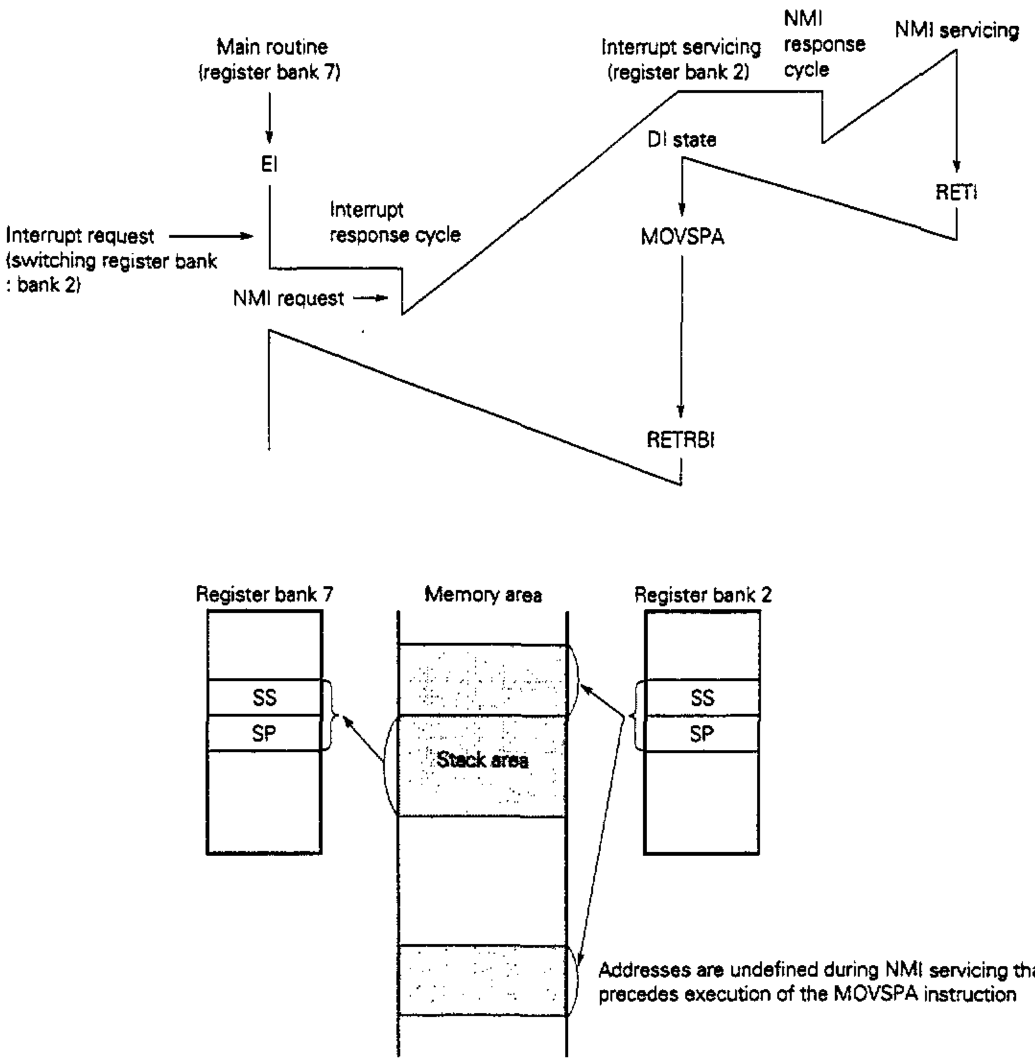




If an NMI request occurs just after response interrupt INTxx is acknowledged by register bank switching, NMI is acknowledged before the INTxx service routine is executed. Thus, when NMI is acknowledged, PS, PC, and PSW are saved in the stack area indicated by SS and SP in the new register bank.

Therefore, in applications where the stack areas of the register banks before and after switching are made contiguous by the MOVSPA instruction, if an NMI request occurs just after response interrupt by register bank switching is acknowledged, the NMI is acknowledged before SS and SP are copied and an undefined memory area is accessed (written) as the stack area. (See **Figure A-2.**)

Figure A-2. Abnormal Operation



A.2.2 Avoidance methods

When using response interrupt and NMI at the same time by register bank switching, avoid the symptom described above by either of the following methods.

(1) Do not use the MOVSPA instruction and set a separate stack area for each switched register bank when an interrupt occurs.

(2) Prevent the stack area from being undefined by providing a spare stack to avoid the symptom described above.

(a) Reserve the area used as the spare stack and preset SS and SP of the switch register bank when an interrupt occurs in the area during the initialization routine.Note

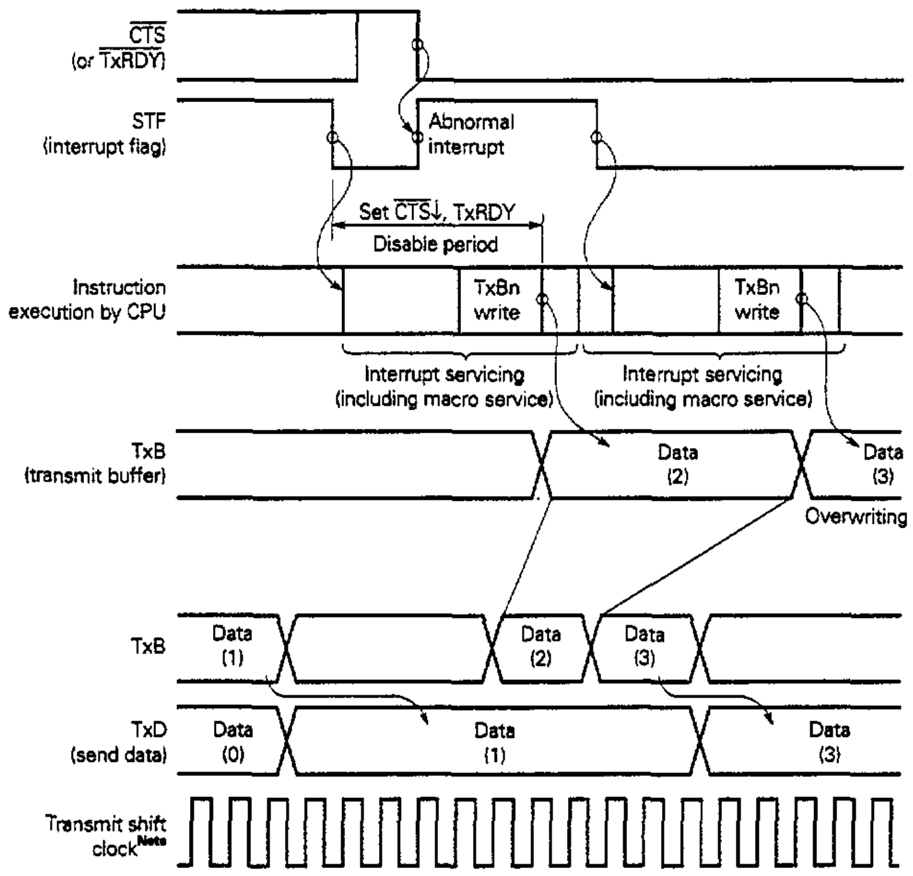
(b) Set the values set in (a) just before the register bank is restored from the new register bank to the former register bank.

Note If (b) is not performed, the stack area becomes undefined again when the second or subsequent interrupt occurs.

A.3 Limitations on Send Data Missing by Transmission Disable Operation during Serial Transmission

A.3.1 Device operation description (symptom)

If the transmission enable state is changed by making the high-to-low transition of $\overline{\text{CTS}}$ input or by setting the TxRDY flag to 1 between the acknowledgment of a transmission completion interrupt request and the next write operation to the transmit buffer (TxB) on the on-chip serial interface, an extra transmission completion interrupt request occurs. (This also applies when macro service is used.) As a result, the send data (one character) written just before may be skipped by the interrupt service routine (or macro service) which overwrites into TxB.



Note When one character consists of 11 bits for transfer (for example, eight data bits, two stop bits, and no parity bit)

A.3.2 Avoidance methods

To write a transmission data to the TxB, read the transmission buffer empty flag (TxBE) in the serial status register (SCS) immediately before the writing and check that the TxB is empty. Note that this prevention method cannot be applied to V25 and V35 as they do not incorporate the transmission buffer empty flag.

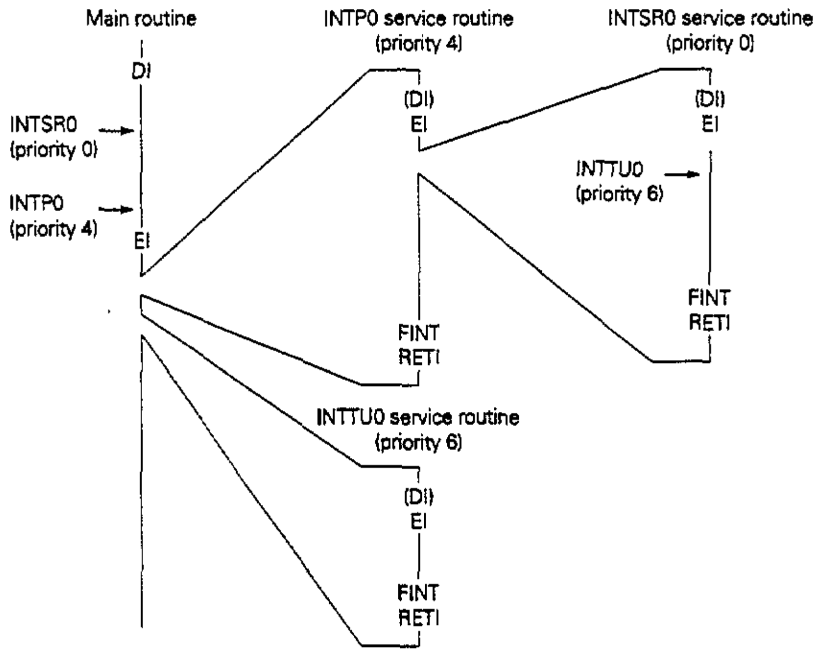
A.4 Cautions on Interrupt Priority Levels and Servicing Sequence

When the CPU state transits from DI to EI, an interrupt having a lower programmable priority among the pending interrupts in the DI state may be acknowledged before an interrupt having a higher programmable priority. (Because the generation timing depends only on the internal timing, the generation timing cannot be externally limited.)

However, if an EI instruction is executed during servicing of an interrupt that has a lower priority level, interrupts that have higher priority levels can be consecutively acknowledged under the multiple servicing control according to the priority levels. For the interrupt that has a lower priority level, execute an EI instruction at the beginning of interrupt servicing to enter the EI state during servicing.

The state should be EI to service the interrupt having the lower priority level. If the DI state is prolonged while servicing the interrupt having the lower priority level, the interrupt requests having higher priority levels that occur in the meantime are held pending during the period, and therefore the priority setting is insignificant. Execute an EI instruction as soon as possible while servicing the interrupt having the lower priority level.

Example of normal interrupt servicing under multiple servicing control



A.5 Caution on Starting Timer Countdown

A.5.1 Device operation description (symptom)

If a countdown is started when the timer register TM value is 0, the following symptoms may occur just after the countdown is started.

- Occurrence of timer interrupt request
- Inversion of TOUT pin level (when ENTO bit = 1)

If such symptoms occur when the interval timer is used, the timer register TM which is counting down can be stopped by clearing (to 0) the TS bit of the TMC register. At that time, if the timer stop timing overlaps with the timing at which TM is set to 0, similar symptoms will occur just after the next countdown is started (the TMC register's TS bit is set to 1).

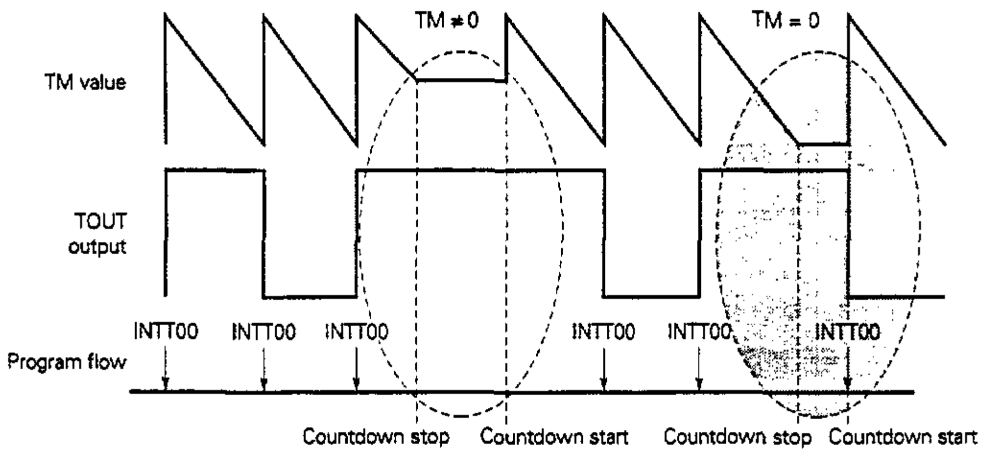
A.5.2 Avoidance method

After stopping the countdown of the TM timer register (by clearing the TS bit), write word data other than 0000H into the TM register. This will suppress any unnecessary occurrence of TOUT pin inversion or a timer interrupt request just after the countdown start and the operation will proceed normally.

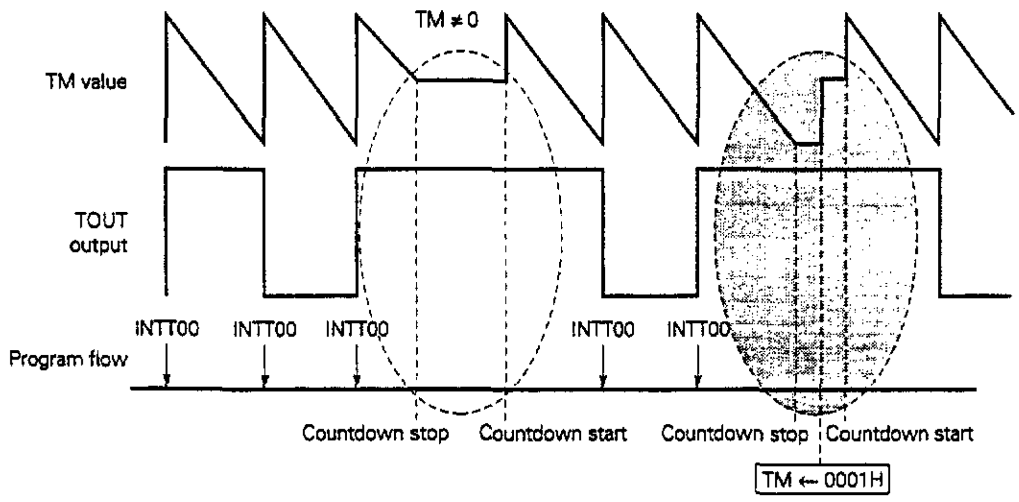
Example

```
MOV   TMC0, 00H
MOV   TMIC0, 00H
MOV   MD0, 0FE0H

MOV   TMC0, 88H     Start countdown

MOV   TMC0, 00H     Stop countdown

MOV   TM0, 0001H    Write a value other than 0000H into TM register

MOV   TMC0, 88H     Start countdown
```

When avoidance method is not performed



When avoidance method is performed

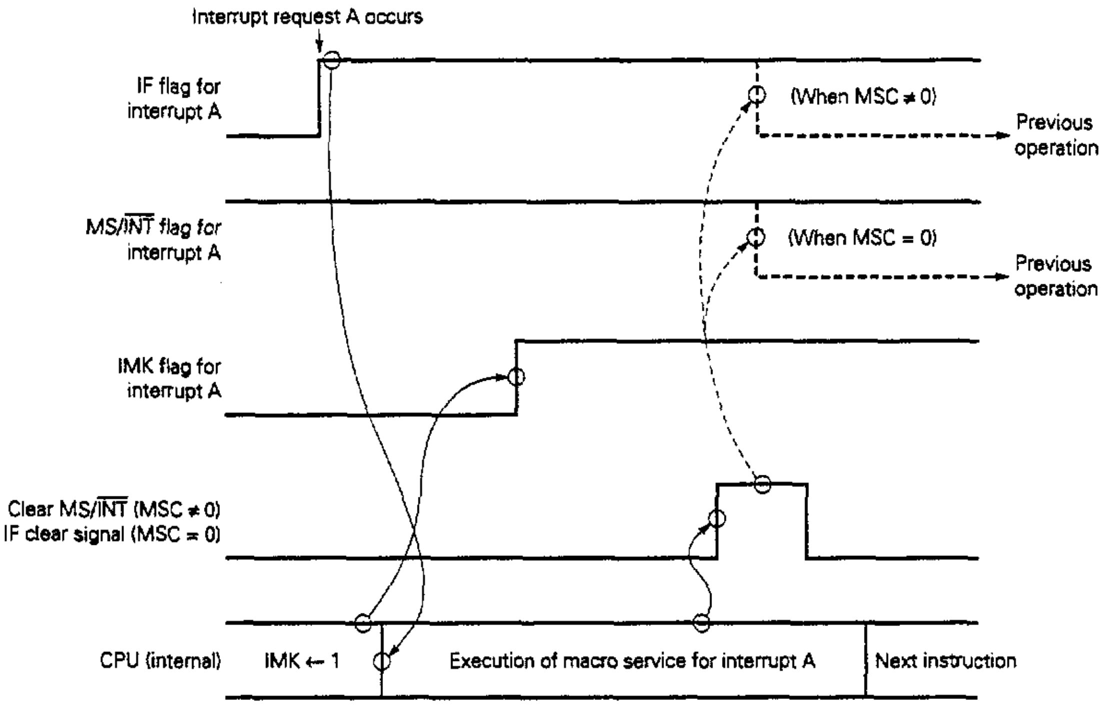


A.6 Limitations on Macro Service Masks

A.6.1 Device operation description (symptom)

Immediately before execution of interrupt A (macro service A) that was set as a macro service response, if an instruction is executed to set macro service A's IMK flag, the IMK flag will be set during execution of the subsequent macro service A. At that time, abnormal execution of macro service A may occur, such as the symptoms described below.

- ***When macro service counter (MSC) ≠ 0***
  Because the IF flag is not cleared after execution of macro service A, macro service A is executed again after the mask is released, even if no interrupt request has occurred.
- **When macro service counter (MSC) = 0**
  Because the MS/$\overline{\text{INT}}$ flag is not cleared after execution of macro service A, a completion interrupt does not occur for macro service A and macro service A is executed again after the mask is released. As a result, MSC is decremented from 0 to 0FFH and macro service is executed each time an interrupt occurs until MSC = 00H.



A.6.2 Avoidance method

Follow the steps shown below when setting or resetting the mask flag set for macro service.

- **Set**

```
DI
CLR1    xxIC, 05H                        ; Reset MS/$\overline{\text{INT}}$ flag
NOP   ┐
NOP   │
NOP   ├ Be sure to insert at least four NOP instructions.
NOP   ┘
SET1    xxIC, 06H                        ; Set IMK flag
EI
```

- **Reset**

```
DI
SET1    xxIC, 05H                        ; Set MS/$\overline{\text{INT}}$ flag
CLR1    xxIC, 06H                        ; Reset IMK flag
EI
```

A.7 Limitations on CVTBD/CVTDB Instructions

A.7.1 Device operation description (symptom)

When the CVTBD or CVTDB instruction is executed immediately after executing one of the instructions listed in Table A-1, data will not be written correctly during the last memory write operation of the instruction (from Table A-1). Instead, undefined data will be written. At that time, only the write data will change and flags will operate normally.

However, this symptom will not occur if internal RAM access is enabled (RAMEN = 1) or if the number of waits for the memory being accessed by the instruction (from Table A-1) is less than two clocks (for the μPD70325) or less than one clock (for the μPD70335).

A.7.2 Avoidance method

Do not execute the CVTBD or CVTDB instruction immediately after executing one of the instructions listed in Table A-1. Instead, execute one or more NOP instructions immediately before the CVTBD or CVTDB instruction.

Example

```
OR      mem, reg
NOP     Be sure to insert at least one NOP instruction.
CVTBD
```

Table A-1. Instructions Subject to Limitations Concerning CVTBD and CVTDB Instructions

Instruction group	Mnemonic	Operand	Instruction group	Mnemonic	Operand
Data transfer instructions	MOV	mem, reg	Bit manipulate instructions	CLR1	mem16, CL
		mem, imm			mem8, imm3
		mem16, sreg			mem16, imm4
	XCH	mem, reg		SET1	mem8, CL
		reg, mem			mem16, CL
Primitive block transfer instructions	MOVBK	dst_block, src_block			mem8, imm3
	STM	dst_block			mem16, imm4
Input/output instructions	OUT	imm8, acc	Shift instructions	SHL	mem, 1
		DW, acc			mem, CL
Primitive input/output instructions	OUTM	DW, src_block			mem, imm8
Add and subtract instructions	ADD	mem, reg		SHR	mem, 1
		mem, imm			mem, CL
	ADDC	mem, reg			mem, imm8
		mem, imm		SHRA	mem, 1
	SUB	mem, reg			mem, CL
		mem, imm			mem, imm8
	SUBC	mem, reg	Rotate instructions	ROL	mem, 1
		mem, imm			mem, CL
BCD operation instructions	ROL4	mem8			mem, imm8
	ROR4	mem8		ROR	mem, 1
Increment and decrement instructions	INC	mem			mem, CL
	DEC	mem			mem, imm8
Complement operation instructions	NOT	mem		ROLC	mem, 1
	NEG	mem			mem, CL
Logical operation instructions	AND	mem, reg			mem, imm8
		mem, imm		RORC	mem, 1
	OR	mem, reg			mem, CL
		mem, imm			mem, imm8
	XOR	mem, reg	Stack manipulate instructions	PUSH	mem16
		mem, imm			reg16
Bit manipulate instructions	NOT1	mem8, CL			sreg
		mem16, CL			PSW
		mem8, imm3			R
		mem16, imm4			imm
	CLR1	mem8, CL		POP	mem16