12 Transputer instruction set summary

12.1 Introduction

The Function Codes table 12.9 (page 63) gives the basic function code set. Where the operand value is less than 16, a single byte encodes the complete instruction. If the operand value is greater than 15, one prefix instruction (*pfix*) is required for each additional four bits of the operand. If the operand is negative the first prefix instruction will be *nfix*. Examples of prefix coding are given in table 12.1.

Mnemonic		Function code	Memory code
ldc	#3	#4	#43
ldc	#35		
is coded as			
pfix	#3	#2	#23
ldc	#5	#4	#45
ldc	#987		
is coded as			
pfix	#9	#2	#29
pfix	#8	#2	#28
ldc	#7	#4	#47
ldc	–31	(*ldc* #FFFFFFE1) (*ldc* #FFE1) †	
is coded as			
nfix	#1	#6	#61
ldc	#1	#4	#41
† IMS T222, IMS T225			

Table 12.1 *prefix* coding

Tables 12.10 to 12.30 (pages 63–70) give details of the operation codes. Where an operation code is less than 16 (e.g. *add*: operation code **05**), the operation can be stored as a single byte comprising the *operate* function code **F** and the operand (**5** in the example). Where an operation code is greater than 15 (e.g. *ladd*: operation code **16**), the *prefix* function code **2** is used to extend the instruction.

Mnemonic		Function code	Memory code
add	(*op. code #5)*		#F5
is coded as			
opr	add	#F	#F5
ladd	(*op. code #16)*		#21F6
is coded as			
pfix	#1	#2	#21
opr	#6	#F	#F6

Table 12.2 *operate* coding

12.1.1 Product identity numbers

The load device identity (*lddevid*) instruction (table 12.10) pushes the device type identity into the A register. Each product is allocated a unique group of numbers for use with the *lddevid* instruction. Product identity numbers are given in table 12.3.

Product	Identity numbers
IMS T425	0 to 9 inclusive
IMS T805	10 to 19 inclusive
IMS T225	40 to 49 inclusive
IMS T400	50 to 59 inclusive

Table 12.3 Product identity numbers

12.1.2 Floating point unit

In the floating point unit (FPU) basic addition, subtraction, multiplication and division operations are performed by single instructions. However, certain less frequently used floating point instructions are selected by a value in register *A* (when allocating registers, this should be taken into account). A *load constant* instruction *ldc* is used to load register *A*; the *floating point entry* instruction *fpentry* then uses this value to select the floating point operation. This pair of instructions is termed a *selector sequence*.

In the Floating Point Operation Codes tables 12.23 to 12.29, a selector sequence code is indicated in the Memory Code column by **s**. The code given in the Operation Code column is the indirection code, the operand for the *ldc* instruction.

The FPU and processor operate concurrently, so the actual throughput of floating point instructions is better than that implied by simply adding up the instruction times. For full details see *Transputer Instruction Set – A Compiler Writer's Guide*.

12.1.3 Notation

The Processor Cycles column refers to the number of periods **TPCLPCL** (refer to **ProcClockOut**) taken by an instruction executing in internal memory. The number of cycles is given for the basic operation only; where the memory code for an instruction is two bytes, the time for the *prefix* function (one cycle) should be added. Some instruction times vary. Where a letter is included in the cycles column it is interpreted from table 12.4.

Ident	Interpretation
b	Bit number of the highest bit set in register **A**. Bit 0 is the least significant bit.
m †	Bit number of the highest bit set in the absolute value of register **A**. Bit 0 is the least significant bit.
n	Number of places shifted.
w	Number of words in the message. Part words are counted as full words. If the message is not word aligned the number of words is increased to include the part words at either end of the message.
p †	Number of words per row.
r †	Number of rows.
† does not apply to IMS T225	

Table 12.4 Instruction set interpretation

The **DEF** column of the tables indicates the descheduling/error features of an instruction as described in table .

Ident	Feature	See section:
D	The instruction is a descheduling point	12.2
E	The instruction will affect the **Error** flag	12.3
F †	The instruction will affect the **FP_Error** flag	12.6
† applies to IMS T805 only		

Table 12.5 Instruction features

12.2 Descheduling points

The instructions in table 12.6 are the only ones at which a process may be descheduled. They are also the ones at which the processor will halt if the **Analyse** pin is asserted (refer to **Analyse** section).

input message	*output message*	*output byte*	*output word*
timer alt wait	*timer input*	*stop on error*	*alt wait*
jump	*loop end*	*end process*	*start process*

Table 12.6 Descheduling point instructions

12.3 Error instructions

The instructions in table 12.7 are the only ones which can affect the *Error* flag directly. Note, however, that the floating point unit error flag *FP_Error* is set by certain floating point instructions (section 12.6), and that *Error* can be set from this flag by *fpcheckerror*.

add	*add constant*	*subtract*	
multiply	*fractional multiply* †	*divide*	*remainder*
long add	*long subtract*	*long divide*	
set error	*testerr*	*fpcheckerror* ‡	
check word	*check subscript from 0*	*check single*	*check count from 1*
† does not apply to IMS T225			
‡ applies to IMS T805 only			

Table 12.7 Error setting instructions

12.4 Debugging support

Table 12.20 (page 67) contains a number of instructions to facilitate the implementation of breakpoints. These instructions overload the operation of *j0*. Normally *j0* is a no-op which might cause descheduling. *Setj0break* enables the breakpointing facilities and causes *j0* to act as a breakpointing instruction. When breakpointing is enabled, *j0* swaps the current **Iptr** and **Wptr** with an **Iptr** and **Wptr** stored above MemStart. The *break* instruction does not cause descheduling, and preserves the state of the registers. It is possible to single step the processor at machine level using these instructions. Refer to *Support for debugging/breakpointing in transputers* (technical note 61) for more detailed information regarding debugger support.

12.5 Block move

The block move instructions (Table 12.21) move any number of bytes from any byte boundary in memory, to any other byte boundary, using the smallest possible number of word read, and word or part-word writes.

A block move instruction can be interrupted by a high priority process. On interrupt, block move is completed to a word boundary, independent of start position. When restarting after interrupt, the last word written is written again. This appears as an unnecessary read and write in the simplest case of word aligned block moves, and may cause problems with FIFOs. This problem can be overcome by incrementing the saved destination (**BregIntSaveLoc**) and source pointer (**CregIntSaveLoc**) values by BytesPerWord during the high priority process.

12.6 Floating point errors (IMS T805 only)

The FPU has its own error flag *FP_Error*. This reflects the state of evaluation within the FPU and is set in circumstances where invalid operations, division by zero or overflow exceptions to the ANSI-IEEE 754-1985 standard would be flagged. *FP_Error* is also set if an input to a floating point operation is infinite or is not a number (NaN). The *FP_Error* flag can be set, tested and cleared without affecting the main *Error* flag, but can also set *Error* when required. Depending on how a program is compiled, it is possible for both unchecked and fully checked floating point arithmetic to be performed.

The instructions in table 12.8 are the only ones which can affect the floating point error flag *FP_Error. Error* is set from this flag by *fpcheckerror* if *FP_Error* is set.

fpadd	*fpsub*	*fpmul*	*fpdiv*
fpldnladdsn	*fpldnladddb*	*fpldnlmulsn*	*fpldnlmuldb*
fpremfirst	*fpusqrtfirst*	*fpgt*	*fpeq*
fpuseterror	*fpuclearerror*	*fptesterror*	
fpuexpincby32	*fpuexpdecby32*	*fpumulby2*	*fpudivby2*
fpur32tor64	*fpur64tor32*	*fpucki32*	*fpucki64*
fprtoi32	*fpuabs*	*fpint*	

Table 12.8 Floating point error setting instructions

12.7 General instructions

The following tables list the complete instruction set which is common to all variants of the transputer. Exceptions are noted at the bottom of each table by † or ‡.

Function Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
0	0X	j	3	jump	D
1	1X	ldlp	1	load local pointer	
2	2X	pfix	1	prefix	
3	3X	ldnl	2	load non–local	
4	4X	ldc	1	load constant	
5	5X	ldnlp	1	load non–local pointer	
6	6X	nfix	1	negative prefix	
7	7X	ldl	2	load local	
8	8X	adc	1	add constant	E
9	9X	call	7	call	
A	AX	cj	2	conditional jump (not taken)	
			4	conditional jump (taken)	
B	BX	ajw	1	adjust workspace	
C	CX	eqc	2	equals constant	
D	DX	stl	1	store local	
E	EX	stnl	2	store non-local	
F	FX	opr	–	operate	

Table 12.9 Function codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
2A	22FA	testpranal	2	test processor analyzing	
3E	23FE	saveh	4	save high priority queue registers	
3D	23FD	savel	4	save low priority queue registers	
18	21F8	sthf	1	store high priority front pointer	
50	25F0	sthb	1	store high priority back pointer	
1C	21FC	stlf	1	store low priority front pointer	
17	21F7	stlb	1	store low priority back pointer	
54	25F4	sttimer	1	store timer	
17C	2127FC	lddevid	1	load device identity	
7E	27FE	ldmemstartval	1	load value of memstart address	

Table 12.10 Processor initialisation operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles		Name	DEF
			16-bit devices	32-bit devices		
46	24F6	and	1	1	and	
4B	24FB	or	1	1	or	
33	23F3	xor	1	1	exclusive or	
32	23F2	not	1	1	bitwise not	
41	24F1	shl	**n**+2	**n**+2	shift left	
40	24F0	shr	**n**+2	**n**+2	shift right	
05	F5	add	1	1	add	E
0C	FC	sub	1	1	subtract	E
53	25F3	mul	23	38	multiply	E
72 †	27F2	fmul		35	fractional multiply (no rounding)	E
				40	fractional multiply (rounding)	E
2C	22FC	div	24	39	divide	E
1F	21FF	rem	21	37	remainder	E
09	F9	gt	2	2	greater than	
04	F4	diff	1	1	difference	
52	25F2	sum	1	1	sum	
08	F8	prod	**b**+4	**b**+4	product for positive register **A**	
08	F8	prod	**m**+5	**m**+5	product for negative register **A**	
† does not apply to IMS T225						

Table 12.11 Arithmetic/logical operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles		Name	DEF
			16-bit devices	32-bit devices		
16	21F6	ladd	2	2	long add	E
38	23F8	lsub	2	2	long subtract	E
37	23F7	lsum	3	3	long sum	
4F	24FF	ldiff	3	3	long diff	
31	23F1	lmul	17	33	long multiply	
1A	21FA	ldiv	19	35	long divide	E
36	23F6	lshl	**n**+3	**n**+3	long shift left (**n**<32) † (**n**<16)	
			n–12	**n**–28	long shift left(**n** 32) † (**n** 16)	
35	23F5	lshr	**n**+3	**n**+3	long shift right (**n**<32) † (**n**<16)	
			n–12	**n**–28	long shift right (**n** 32) † (**n** 16)	
19	21F9	norm	**n**+5	**n**+5	normalise (**n**<32) † (**n**<16)	
			n–10	**n**–26	normalise (**n** 32) † (**n** 16)	
			3	3	normalise (**n**=64) † (**n**=32)	
† for IMS T225						

Table 12.12 Long arithmetic operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
00	F0	rev	1	reverse	
3A	23FA	xword	4	extend to word	
56	25F6	cword	5	check word	E
1D	21FD	xdble	2	extend to double	
4C	24FC	csngl	3	check single	E
42	24F2	mint	1	minimum integer	
5A	25FA	dup	1	duplicate top of stack	
79	27F9	pop	1	pop processor stack	

Table 12.13 General operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles		Name	DEF
			16-bit devices	32-bit devices		
02	F2	bsub	1	1	byte subscript	
0A	FA	wsub	2	2	word subscript	
81 †	28F1	wsubdb	3	3	form double word subscript	
34	23F4	bcnt	2	2	byte count	
3F	23FF	wcnt	4	5	word count	
01	F1	lb	5	5	load byte	
3B	23FB	sb	4	4	store byte	
4A	24FA	move	2**w**+8	2**w**+8	move message	
† does not apply to IMS T225						

Table 12.14 Indexing/array operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
22	22F2	ldtimer	2	load timer	
2B	22FB	tin	30	timer input (time future)	D
			4	timer input (time past)	D
4E	24FE	talt	4	timer alt start	
51	25F1	taltwt	15	timer alt wait (time past)	D
			48	timer alt wait (time future)	D
47	24F7	enbt	8	enable timer	
2E	22FE	dist	23	disable timer	

Table 12.15 Timer handling operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
07	F7	in	2**w**+19	input message	D
0B	FB	out	2**w**+19	output message	D
0F	FF	outword	23	output word	D
0E	FE	outbyte	23	output byte	D
43	24F3	alt	2	alt start	
44	24F4	altwt	5	alt wait (channel ready)	D
			17	alt wait (channel not ready)	D
45	24F5	altend	4	alt end	
49	24F9	enbs	3	enable skip	
30	23F0	diss	4	disable skip	
12	21F2	resetch	3	reset channel	
48	24F8	enbc	7	enable channel (ready)	
			5	enable channel (not ready)	
2F	22FF	disc	8	disable channel	

Table 12.16 Input/output operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
20	22F0	ret	5	return	
1B	21FB	ldpi	2	load pointer to instruction	
3C	23FC	gajw	2	general adjust workspace	
06	F6	gcall	4	general call	
21	22F1	lend	10	loop end (loop)	D
			5	loop end (exit)	D

Table 12.17 Control operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
0D	FD	startp	12	start process	
03	F3	endp	13	end process	D
39	23F9	runp	10	run process	
15	21F5	stopp	11	stop process	
1E	21FE	ldpri	1	load current priority	

Table 12.18 Scheduling operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
13	21F3	csub0	2	check subscript from 0	E
4D	24FD	ccnt1	3	check count from 1	E
29	22F9	testerr	2	test error false and clear (no error)	
			3	test error false and clear (error)	
10	21F0	seterr	1	set error	E
55	25F5	stoperr	2	stop on error (no error)	D
57	25F7	clrhalterr	1	clear halt–on–error	
58	25F8	sethalterr	1	set halt–on–error	
59	25F9	testhalterr	2	test halt–on–error	

Table 12.19 Error handling operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
0	00	jump 0	3	jump 0 (break not enabled)	D
			11	jump 0 (break enabled, high priority)	
			13	jump 0 (break enabled, low priority)	
B1	2BF1	break	9	break (high priority)	
			11	break (low priority)	
B2	2BF2	clrj0break	1	clear jump 0 break enable flag	
B3	2BF3	setj0break	1	set jump 0 break enable flag	
B4	2BF4	testj0break	2	test jump 0 break enable flag set	
7A	27FA	timerdisableh	1	disable high priority timer interrupt	
7B	27FB	timerdisablel	1	disable low priority timer interrupt	
7C	27FC	timerenableh	6	enable high priority timer interrupt	
7D	27FD	timerenablel	6	enable low priority timer interrupt	

Table 12.20 Debugger support codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
5B †	25FB	move2dinit	8	initialise data for 2D block move	
5C †	25FC	move2dall	(2**p**+23)***r**	2D block copy	
5D †	25FD	move2dnonzero	(2**p**+23)***r**	2D block copy non-zero bytes	
5E †	25FE		(2**p**+23)***r**	2D block copy zero bytes	
† does not apply to IMS T225					

Table 12.21 2D block move operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
74	27F4	crcword	35	calculate crc on word	
75	27F5	crcbyte	11	calculate crc on byte	
76	27F6	bitcnt	**b**+2	count bits set in word	
77	27F7	bitrevword	36	reverse bits in word	
78	27F8	bitrevnbits	**n**+4	reverse bottom n bits in word	

Table 12.22 CRC and bit operation codes

12.8 Floating point instructions

12.9 Floating point instructions for IMS T805 only

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
8E	28FE	fpldnlsn	2	fp load non-local single	
8A	28FA	fpldnldb	3	fp load non-local double	
86	28F6	fpldnlsni	4	fp load non-local indexed single	
82	28F2	fpldnldbi	6	fp load non-local indexed double	
9F	29FF	fpldzerosn	2	load zero single	
A0	2AF0	fpldzerodb	2	load zero double	
AA	2AFA	fpldnladdsn	8/11	fp load non local & add single	F
A6	2AF6	fpldnladddb	9/12	fp load non local & add double	F
AC	2AFC	fpldnlmulsn	13/20	fp load non local & multiply single	F
A8	2AF8	fpldnlmuldb	21/30	fp load non local & multiply double	F
88	28F8	fpstnlsn	2	fp store non-local single	
84	28F4	fpstnldb	3	fp store non-local double	
9E	29FE	fpstnli32	4	store non-local int32	
Processor cycles are shown as **Typical/Maximum** cycles.					

Table 12.23 Floating point load/store operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
AB	2AFB	fpentry	1	floating point unit entry	
A4	2AF4	fprev	1	fp reverse	
A3	2AF3	fpdup	1	fp duplicate	

Table 12.24 Floating point general operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
22	**s**	fpurn	1	set rounding mode to round nearest	
06	**s**	fpurz	1	set rounding mode to round zero	
04	**s**	fpurp	1	set rounding mode to round positive	
05	**s**	fpurm	1	set rounding mode to round minus	

Table 12.25 Floating point rounding operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
83	28F3	fpchkerror	1	check fp error	E
9C	29FC	fptesterror	2	test fp error false and clear	F
23	**s**	fpuseterror	1	set fp error	F
9C	**s**	fpuclearerror	1	clear fp error	F

Table 12.26 Floating point error operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
94	29F4	fpgt	4/6	fp greater than	F
95	29F5	fpeq	3/5	fp equality	F
92	29F2	fpordered	3/4	fp orderability	
91	29F1	fpnan	2/3	fp NaN	
93	29F3	fpnotfinite	2/2	fp not finite	
0E	**s**	fpuchki32	3/4	check in range of type int32	F
0F	**s**	fpuchki64	3/4	check in range of type int64	F
Processor cycles are shown as **Typical/Maximum** cycles.					

Table 12.27 Floating point comparison operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
07	**s**	fpur32tor64	3/4	real32 to real64	F
08	**s**	fpur64tor32	6/9	real64 to real32	F
9D	29FD	fprtoi32	7/9	real to int32	F
96	29F6	fpi32tor32	8/10	int32 to real32	
98	29F8	fpi32tor64	8/10	int32 to real64	
9A	29FA	fpb32tor64	8/8	bit32 to real64	
0D	**s**	fpunoround	2/2	real64 to real32, no round	
A1	2AF1	fpint	5/6	round to floating integer	F
Processor cycles are shown as **Typical/Maximum** cycles.					

Table 12.28 Floating point conversion operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles		Name	DEF
			Single	Double		
87	28F7	fpadd	6/9	6/9	fp add	F
89	28F9	fpsub	6/9	6/9	fp subtract	F
8B	28FB	fpmul	11/18	18/27	fp multiply	F
8C	28FC	fpdiv	16/28	31/43	fp divide	F
0B	**s**	fpuabs	2/2	2/2	fp absolute	F
8F	28FF	fpremfirst	36/46	36/46	fp remainder first step	F
90	29F0	fpremstep	32/36	32/36	fp remainder iteration	
01	**s**	fpusqrtfirst	27/29	27/29	fp square root first step	F
02	**s**	fpusqrtstep	42/42	42/42	fp square root step	
03	**s**	fpusqrtlast	8/9	8/9	fp square root end	
0A	**s**	fpuexpinc32	6/9	6/9	multiply by 2^{32}	F
09	**s**	fpuexpdec32	6/9	6/9	divide by 2^{32}	F
12	**s**	fpumulby2	6/9	6/9	multiply by 2.0	F
11	**s**	fpudivby2	6/9	6/9	divide by 2.0	F
Processor cycles are shown as **Typical/Maximum** cycles.						

Table 12.29 Floating point arithmetic operation codes

12.10 Floating point instructions for IMS T400 and IMS T425 only

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
73	27F3	cflerr	3	check floating point error	E
9C	29FC	fptesterr	1	load value true (FPU not present)	
63	26F3	unpacksn	15	unpack single length fp number	
6D	26FD	roundsn	12/15	round single length fp number	
6C	26FC	postnormsn	5/30	post–normalise correction of single length fp number	
71	27F1	ldinf	1	load single length infinity	
Processor cycles are shown as **Typical/Maximum** cycles.					

Table 12.30 Floating point support operation codes