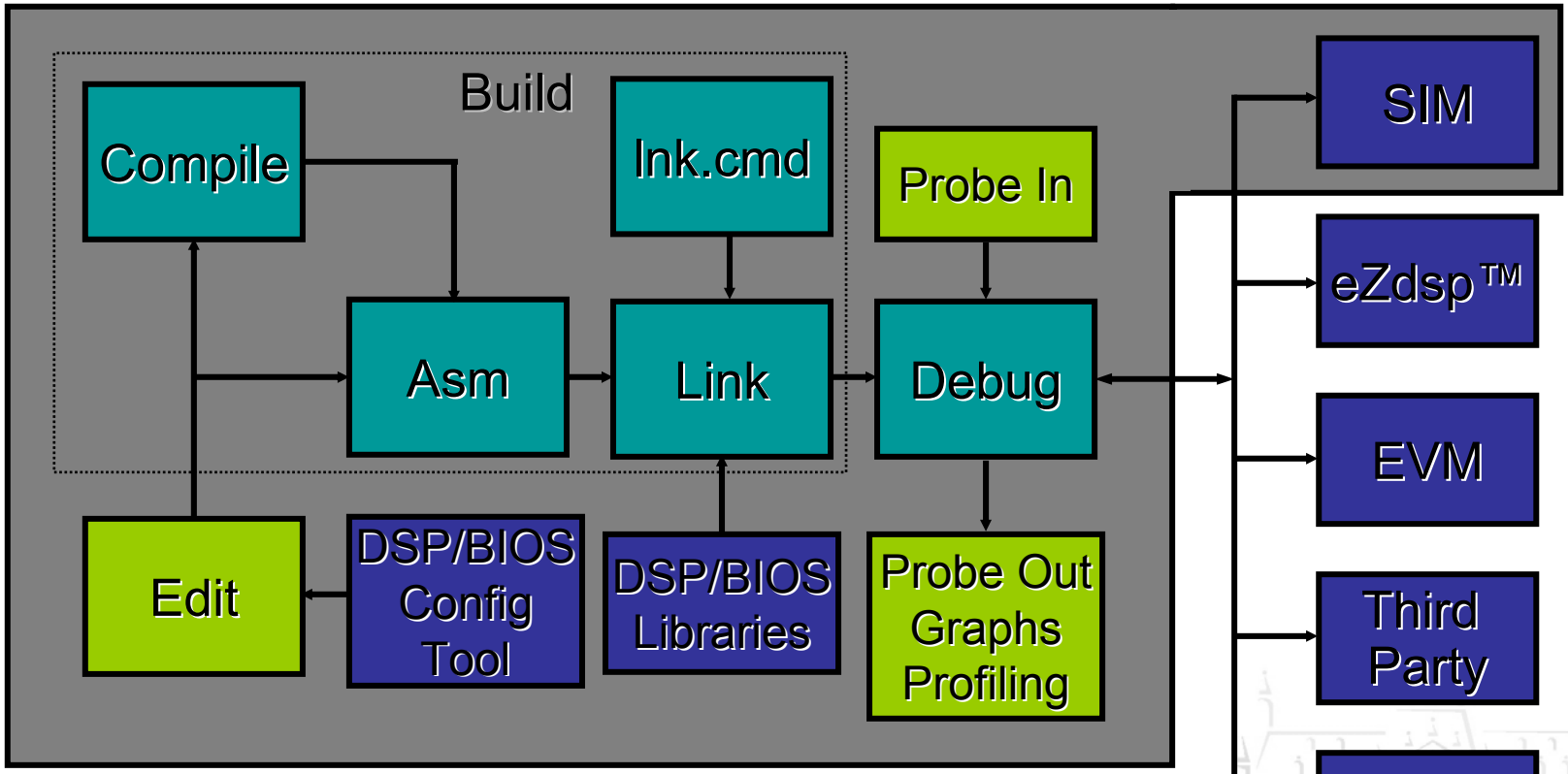
Code Composer Studio



- Code Composer Studio vsebuje:
  - Integrirani Edit/Debug GUI
  - Orodja za generiranje kode (Code Generation)
  - DSP/BIOS

Code Composer Studio® IDE

Project Manager:

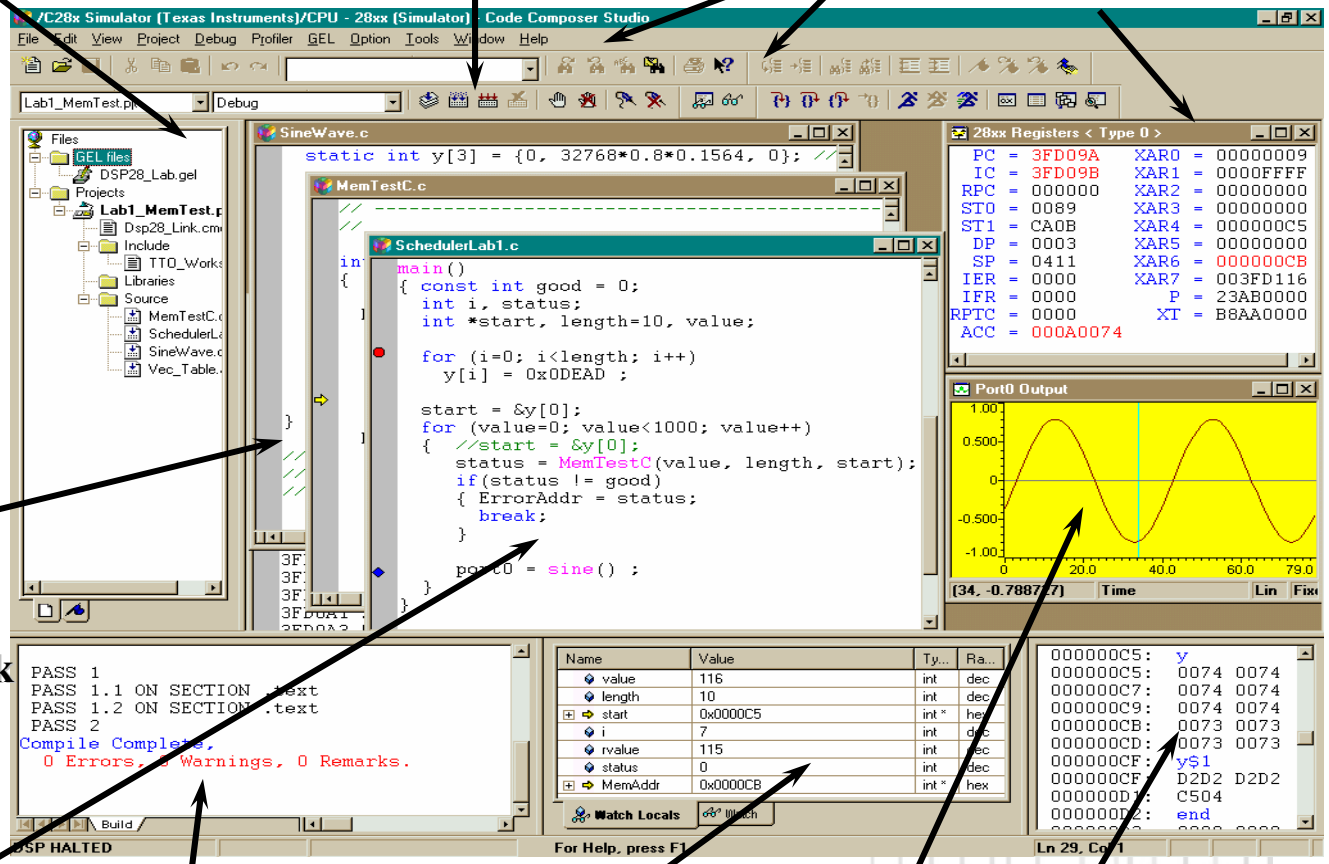
- Izvirne in objektne (Source & object) datoteke
- Povezave med datotekami
- Opcije prevajalnika, zbirnika in povezovalnika (Compiler, Assembler & Linker)

Polno C/C++ & Assembly razhroščevanje (debugging):

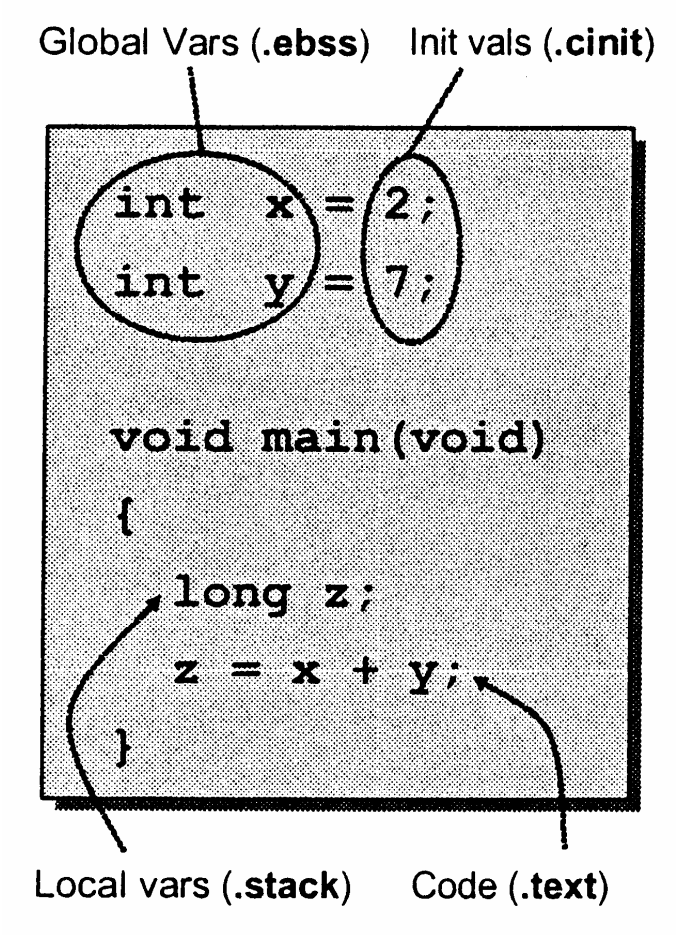
- C & ASM izvor
- Mešani način
- Dizasembler (disassembly - patch)
- Postavitev prekinitvenih točk (break points)
- Postavitev sondirnih točk
- (probe points)

Productive Editor:

- Strukturna razširitev



Sekcije*



- Vsak program v C jeziku je sestavljen iz sekcij
- Po definiciji, sekcije so označene s piko “.”
- C prevajalnik vsebuje prednastavljena imena za inicializirane in neinicializirane sekcij

*C28x DSP Design Workshop, Student guide, Rev. 6.2TTO, okt. 2005

Nazivi sekcij v C prevajalniku*

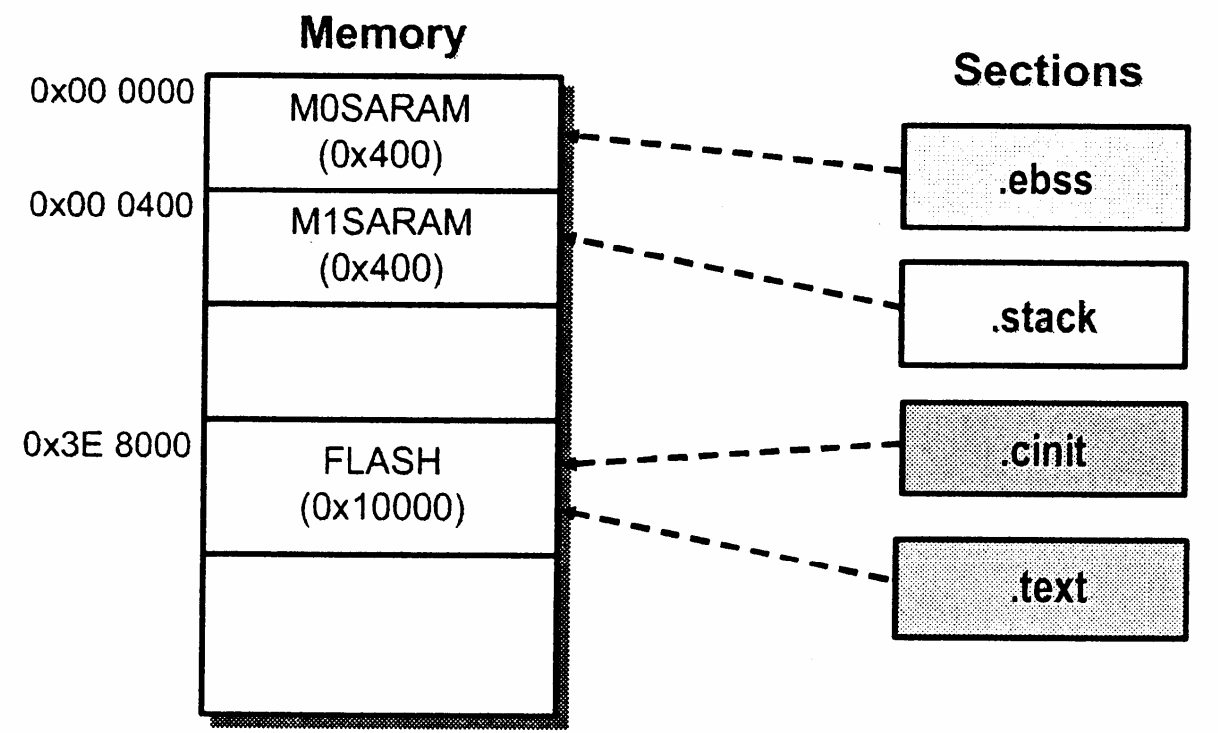
Initialized Sections

Name	Description	Link Location
.text	code	program
.cinit	initialized global and static variables	program
.econst	constant data (e.g. const int k = 3;)	data
.switch	tables for switch statements	data (prog w/ -mt)
.pinit	tables for global constructors (C++)	program

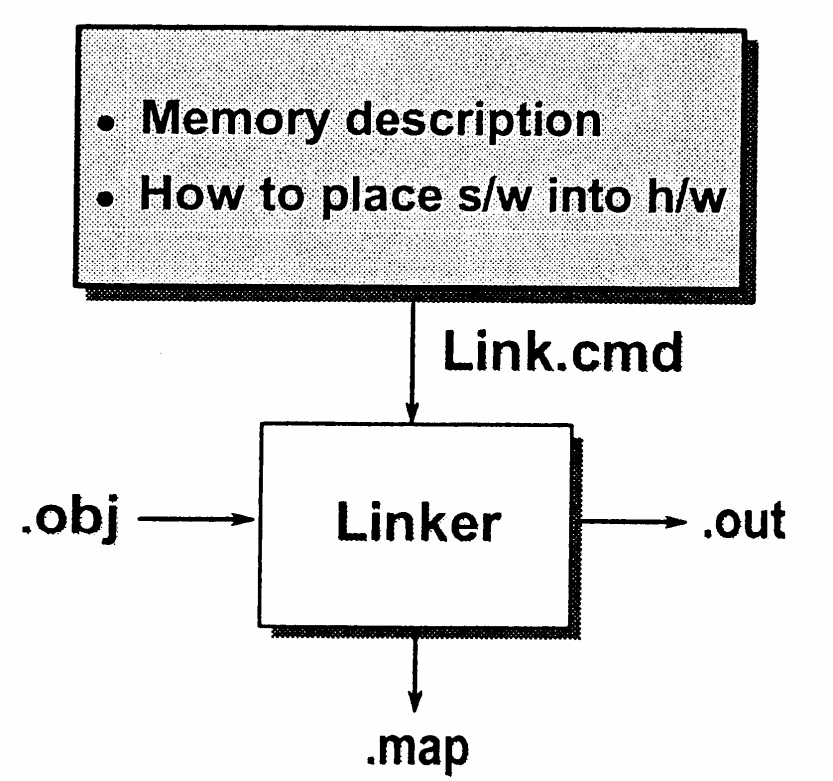
Uninitialized Sections

Name	Description	Link Location
.ebss	global & static variables	data
.stack	stack space	low 64K data
.esysmem	memory for far malloc functions	data

Lociranje sekcij v pomniliniku*



Povezovanje (linking)*



Linker command File*

```
MEMORY
{
   PAGE 0:          /* Program Space */
    FLASH:     org = 0x3E8000,  len = 0x10000

   PAGE 1:          /* Data Space */
    M0SARAM:   org = 0x000000,  len = 0x400
    M1SARAM:   org = 0x000400,  len = 0x400
}
```

Linker command File (določanje sekcij)*

```
MEMORY
{
   PAGE 0:          /* Program Space */
    FLASH:     org = 0x3E8000,  len = 0x10000

   PAGE 1:          /* Data Space */
    M0SARAM:   org = 0x000000,  len = 0x400
    M1SARAM:   org = 0x000400,  len = 0x400
}
SECTIONS
{
    .text:>         FLASH       PAGE 0
    .ebss:>         M0SARAM     PAGE 1
    .cinit:>        FLASH       PAGE 0
    .stack:>        M1SARAM     PAGE 1
}
```

Označevanje simbolov

XARn	XAR0 to XAR7 registers
ARn, ARm	Lower 16-bits of XAR0 to XAR7 registers
ARnH	Upper 16-bits of XAR0 to XAR7 registers
ARPn	3-bit auxiliary register pointer, ARP0 to ARP7
ARP0	points to XAR0 and ARP7 points to XAR7
AR(ARP)	Lower 16-bits of auxiliary register pointed to by ARP
XAR(ARP)	Auxiliary registers pointed to by ARP
AX	Accumulator high (AH) and low (AL) registers
#	Immediate operand
PM	Product shift mode (+4,1,0,-1,-2,-3,-4,-5,-6)
PC	Program counter
~	Bitwise compliment
[loc16]	Contents of 16-bit location
0:[loc16]	Contents of 16-bit location, zero extended
S:[loc16]	Contents of 16-bit location, sign extended
[loc32]	Contents of 32-bit location
0:[loc32]	Contents of 32-bit location, zero extended
S:[loc32]	Contents of 32-bit location, sign extended
7bit	7-bit immediate value
0:7bit	7-bit immediate value, zero extended
S:7bit	7-bit immediate value, sign extended
8bit	8-bit immediate value

S:8bit	8-bit immediate value, sign extended
10bit	10-bit immediate value
0:10bit	10-bit immediate value, zero extended
16bit	16-bit immediate value
0:16bit	16-bit immediate value, zero extended
S:16bit	16-bit immediate value, sign extended
22bit	22-bit immediate value
0:22bit	22-bit immediate value, zero extended
LSb	Least Significant bit
LSB	Least Significant Byte
LSW	Least Significant Word
MSb	Most Significant bit
MSB	Most Significant Byte
MSW	Most Significant Word
OBJ	OBJMODE bit state for which instruction is valid
N	Repeat count (N = 0,1,2,3,4,5,6,7,....)
{ }	Optional field
=	Assignment
==	Equivalent to

XARn Register Operations (XAR0-XAR7)

ADDB	XARn,#7bit	Add 7-bit constant to auxiliary register
ADRK	#8bit	Add 8-bit constant to current auxiliary register
CMPR	0/1/2/3	Compare auxiliary registers
MOV	AR6/7,loc16	Load auxiliary register
MOV	loc16,ARn	Store 16-bit auxiliary register
MOV	XARn,PC	Save the current program counter
MOVB	XARn,#8bit	Load auxiliary register with 8-bit value
MOVB	AR6/7,#8bit	Load auxiliary register with an 8-bit constant
MOVL	XARn,loc32	Load 32-bit auxiliary register
MOVL	loc32,XARn	Store 32-bit auxiliary register
MOVL	XARn,#22bit	Load 32-bit auxiliary register with constant value
MOVZ	ARn,loc16	Load lower half of XARn and clear upper half
SBRK	#8bit	Subtract 8-bit constant from current auxiliary register
SUBB	XARn,#7bit	Subtract 7-bit constant from auxiliary register

DP Register Operations

MOV	DP,#10bit	Load data-page pointer
MOVW	DP,#16bit	Load the entire data page
MOVZ	DP,#10bit	Load data page and clear high bits

SP Register Operations

ADDB	SP,#7bit	Add 7-bit constant to stack pointer
POP	ACC	Pop ACC register from stack
POP	AR1:AR0	Pop AR1 & AR0 registers from stack
POP	AR1H:AR0H	Pop AR1H & AR0H registers from stack

SP Register Operations (nadaljevanje)

POP	AR3:AR2	Pop AR3 & AR2 registers from stack
POP	AR5:AR4	Pop AR5 & AR4 registers from stack
POP	DBGIER	Pop DBGIER register from stack
POP	DP:ST1	Pop DP & ST1 registers on stack
POP	DP	Pop DP register from stack
POP	IFR	Pop IFR register from stack
POP	loc16	Pop .loc16. data from stack
POP	P	Pop P register from stack
POP	RPC	Pop RPC register from stack
POP	ST0	Pop ST0 register from
POP	ST1	Pop ST1 register from stack
POP	T:ST0	Pop T & ST0 registers from stack 9
POP	XT	Pop XT register from stack
POP	XARn	Pop auxiliary register from stack
PUSH	ACC	Push ACC register on stack
PUSH	ARn:ARn	Push ARn & ARn registers on stack
PUSH	AR1H:AR0H	Push AR1H & AR0H registers on stack
PUSH	DBGIER	Push DBGIER register on stack
PUSH	DP:ST1	Push DP & ST1 registers on stack
PUSH	DP	Push DP register on stack
PUSH	IFR	Push IFR register on stack
PUSH	loc16	Push .loc16. data on stack
PUSH	P	Push P register on stack
PUSH	RPC	Push RPC register on stack
PUSH	ST0	Push ST0 register on stack 6-294
PUSH	ST1	Push ST1 register on stack 6-295

SP Register Operations (nadaljevanje)

PUSH	T:ST0	Push T & ST0 registers on stack
PUSH	XT	Push XT register on stack
PUSH	XARn	Push auxiliary register on stack
SUBB	SP,#7bit	Subtract 7-bit constant from the stack pointer

AX Register Operations (AH, AL)

ADD	AX,loc16	Add value to AX
ADD	loc16,AX	Add AX to specified location
ADDB	AX,#8bit	Add 8-bit constant to AX
AND	AX,loc16,#16bit	Bitwise AND
AND	AX,loc16	Bitwise AND
AND	loc16,AX	Bitwise AND
ANDB	AX,#8bit	Bitwise AND 8-bit value
ASR	AX,1..16	Arithmetic shift right
ASR	AX,T	Arithmetic shift right by T(3:0) = 0...15
CMP	AX,loc16	Compare
CMPB	AX,#8bit	Compare 8-bit value
FLIP	AX	Flip order of bits in AX register
LSL	AX,1..16	Logical shift left
LSL	AX,T	Logical shift left by T(3:0) = 0...15
LSR	AX,1..16	Logical shift right
LSR	AX,T	Logical shift right by T(3:0) = 0..15
MAX	AX,loc16	Find the maximum
MIN	AX,loc16	Find the minimum
MOV	AX,loc16	Load AX
MOV	loc16,AX	Store AX
MOV	loc16,AX,COND	Store AX register conditionally

AX Register Operations (AH, AL) (nadaljevanje)

MOVB	AX,#8bit	Load AX with 8-bit constant 6-189
MOVB	AX.LSB,loc16	Load LSB of AX reg, MSB = 0x00 6-190
MOVB	AX.MSB,loc16	Load MSB of AX reg, LSB = unchanged 6-192
MOVB	loc16,AX.LSB	Store LSB of AX reg 6-196
MOVB	loc16,AX.MSB	Store MSB of AX reg 6-198
NEG	AX	Negate AX register 6-245
NOT	AX	Complement AX register 6-256
OR	AX,loc16	Bitwise OR 6-259
OR	loc16,AX	Bitwise OR 6-263
ORB	AX,#8bit	Bitwise OR 8-bit value 6-264
SUB	AX,loc16	Subtract specified location from AX 6-338
SUB	loc16,AX	Subtract AX from specified location 6-339
SUBR	loc16,AX	Reverse-subtract specified location from AX 6-354
SXTB	AX	Sign extend LSB of AX reg into MSB
XOR	AX,loc16	Bitwise exclusive OR 6-384
XORB	AX,#8bit	Bitwise exclusive OR 8-bit value 6-387
XOR	loc16,AX	Bitwise exclusive OR 6-385

16-Bit ACC Register Operations

ADD	ACC,loc16 {<< 0..16}	Add value to accumulator 6-25
ADD	ACC,#16bit {<< 0..15}	Add value to accumulator 6-22
ADD	ACC,loc16 << T	Add shifted value to accumulator 6-24
ADDB	ACC,#8bit	Add 8-bit constant to accumulator 6-30
ADDCU	ACC,loc16	Add unsigned value plus carry to accumulator 6-35
ADDU	ACC,loc16	Add unsigned value to accumulator 6-39
AND	ACC,loc16	Bitwise AND 6-44
AND	ACC,#16bit {<< 0..16}	Bitwise AND

16-Bit ACC Register Operations (nadaljevanje)

MOV	ACC,loc16 {<< 0..16}	Load accumulator with shift 6-159
MOV	ACC,#16bit {<< 0..15}	Load accumulator with shift 6-159
MOV	loc16,ACC << 1..8	Save low word of shifted accumulator 6-167
MOV	ACC,loc16 << T	Load accumulator with shift 6-158
MOVB	ACC,#8bit	Load accumulator with 8-bit value 6-187
MOVH	loc16,ACC << 1..8	Save high word of shifted accumulator 6-202
MOVU	ACC,loc16	Load accumulator with unsigned word 6-220
SUB	ACC,loc16 << T	Subtract shifted value from accumulator 6-335
SUB	ACC,loc16 {<< 0..16}	Subtract shifted value from accumulator 6-333
SUB	ACC,#16bit {<< 0..15}	Subtract shifted value from accumulator 6-337
SUBB	ACC,#8bit	Subtract 8-bit value 6-340
SBBU	ACC,loc16	Subtract unsigned value plus inverse borrow 6-317
SUBU	ACC,loc16	Subtract unsigned 16-bit value 6-356
OR	ACC,loc16	Bitwise OR 6-257
OR	ACC,#16bit {<< 0..16}	Bitwise OR 6-258
XOR	ACC,loc16	Bitwise exclusive OR 6-382
XOR	ACC,#16bit {<< 0..16}	Bitwise exclusive OR 6-383
ZALR	ACC,loc16	Zero AL and load AH with rounding 6-394

32-Bit ACC Register Operations

ABS	ACC	Absolute value of accumulator 6-19
ABSTC	ACC	Absolute value of accumulator and load TC 6-20
ADDL	ACC,loc32	Add 32-bit value to accumulator 6-36
ADDL	loc32,ACC	Add accumulator to specified location 6-38
ADDCL	ACC,loc32	Add 32-bit value plus carry to accumulator 6-34
ADDUL	ACC,loc32	Add 32-bit unsigned value to accumulator 6-41
ADDL	ACC,P << PM	Add shifted P to accumulator

32-Bit ACC Register Operations (nadaljevanje)

ASRL	ACC,T	Arithmetic shift right of accumulator by T(4:0)
CMPL	ACC,loc32	Compare 32-bit value
CMPL	ACC,P << PM	Compare 32-bit value
CSB	ACC	Count sign bits
LSL	ACC,1..16	Logical shift left 1 to 16 places
LSL	ACC,T	Logical shift left by T(3:0) = 0...15
LSRL	ACC,T	Logical shift right by T(4:0)
LSLL	ACC,T	Logical shift left by T(4:0)
MAXL	ACC,loc32	Find the 32-bit maximum
MINL	ACC,loc32	Find the 32-bit minimum
MOVL	ACC,loc32	Load accumulator with 32 bits
MOVL	loc32,ACC	Store 32-bit accumulator
MOVL	P,ACC	Load P from the accumulator
MOVL	ACC,P << PM	Load the accumulator with shifted P
MOVL	loc32,ACC,COND	Store ACC conditionally
NORM	ACC,XARn++/--	Normalize ACC and modify selected auxiliary register.
NORM	ACC,*ind	C2XLP compatible Normalize ACC operation
NEG	ACC	Negate ACC
NEGTC	ACC	If TC is equivalent to 1, negate ACC
NOT	ACC	Complement ACC
ROL	ACC	Rotate ACC left
ROR	ACC	Rotate ACC right
SAT	ACC	Saturate ACC based on OVC value
SFR	ACC,1..16	Shift accumulator right by 1 to 16 places
SFR	ACC,T	Shift accumulator right by T(3:0) = 0...15
SUBBL	ACC,loc32	Subtract 32-bit value plus inverse borrow

32-Bit ACC Register Operations (nadaljevanje)

SUBCU	ACC,loc16	Subtract conditional 16-bit value
SUBCUL	ACC,loc32	Subtract conditional 32-bit value
SUBL	ACC,loc32	Subtract 32-bit value
SUBL	loc32,ACC	Subtract 32-bit value
SUBL	ACC,P << PM	Subtract 32-bit value
SUBRL	loc32,ACC	Reverse-subtract specified location from ACC
SUBUL	ACC,loc32	Subtract unsigned 32-bit value
TEST	ACC	Test for accumulator equal to zero

64-Bit ACC:P Register Operations

ASR64	ACC:P,#1..16	Arithmetic shift right of 64-bit value
ASR64	ACC:P,T	Arithmetic shift right of 64-bit value by T(5:0)
CMP64	ACC:P	Compare 64-bit value
LSL64	ACC:P,1..16	Logical shift left 1 to 16 places
LSL64	ACC:P,T	64-bit logical shift left by T(5:0)
LSR64	ACC:P,#1..16	64-bit logical shift right by 1 to 16 places
LSR64	ACC:P,T	64-bit logical shift right by T(5:0)
NEG64	ACC:P	Negate ACC:P
SAT64	ACC:P	Saturate ACC:P based on OVC value

P or XT Register Operations (P, PH, PL, XT, T, TL)

ADDUL	P,loc32	Add 32-bit unsigned value to P
MAXCUL	P,loc32	Conditionally find the unsigned maximum
MINCUL	P,loc32	Conditionally find the unsigned minimum
MOV	PH,loc16	Load the high half of the P register
MOV	PL,loc16	Load the low half of the P register
MOV	loc16,P	Store lower half of shifted P register
MOV	T,loc16	Load the upper half of the XT register

P or XT Register Operations (P, PH, PL, XT, T, TL) (nadaljevanje)

MOV	loc16,T	Store the T register
MOV	TL,#0	Clear the lower half of the XT register
MOVA	T,loc16	Load the T register and add the previous product
MOVAD	T,loc16	Load T register
MOVDL	XT,loc32	Store XT and load new XT
MOVH	loc16,P	Save the high word of the P register
MOVL	P,loc32	Load the P register
MOVL	loc32,P	Store the P register
MOVL	XT,loc32	Load the XT register
MOVL	loc32,XT	Store the XT register
MOVP	T,loc16	Load the T register and store P in the accumulator
MOVS	T,loc16	Load T and subtract P from the accumulator
MOVX	TL,loc16	Load lower half of XT with sign extension
SUBUL	P,loc32	Subtract unsigned 32-bit value

16x16 Multiply Operations

DMAC	ACC:P,loc32,*XAR7/++	16-bit dual multiply and accumulate
MAC	P,loc16,0:pma	Multiply and accumulate
MAC	P,loc16,*XAR7/++	Multiply and Accumulate
MPY	P,T,loc16	16 X 16 multiply
MPY	P,loc16,#16bit	16 X 16-bit multiply
MPY	ACC,T,loc16	16 X 16-bit multiply
MPY	ACC,loc16,#16bit	16 X 16-bit multiply
MPYA	P,loc16,#16bit	16 X 16-bit multiply and add previous product
MPYA	P,T,loc16	16 X 16-bit multiply and add previous product
MPYB	P,T,#8bit	Multiply signed value by unsigned 8-bit constant
MPYS	P,T,loc16	16 X 16-bit multiply and subtract

16x16 Multiply Operations (nadaljevanje)

MPYB	ACC,T,#8bit	Multiply by 8-bit constant
MPYU	ACC,T,loc16	16 X 16-bit unsigned multiply
MPYU	P,T,loc16	Unsigned 16 X 16 multiply
MPYXU	P,T,loc16	Multiply signed value by unsigned value
MPYXU	ACC,T,loc16	Multiply signed value by unsigned value
SQRA	loc16	Square value and add P to accumulator
SQRS	loc16	Square value and subtract from accumulator
XMAC	P,loc16,*(pma)	C2xLP source-compatible multiply and accumulate
XMACD	P,loc16,*(pma)	C2xLP source-compatible multiply and accumulate with data move

32x32 Multiply Operations

IMACL	P,loc32,*XAR7/++	Signed 32 X 32-bit multiply and accumulate (lower half)
IMPYAL	P,XT,loc32	Signed 32-bit multiply (lower half) and add previous P
IMPYL	P,XT,loc32	Signed 32 X 32-bit multiply (lower half)
IMPYL	ACC,XT,loc32	Signed 32 X 32-bit multiply (lower half)
IMPYSL	P,XT,loc32	Signed 32-bit multiply (lower half) and subtract P
IMPYXUL	P,XT,loc32	Signed 32 X unsigned 32-bit multiply (lower half)
QMACL	P,loc32,*XAR7/++	Signed 32 X 32-bit multiply and accumulate (upper half)
QMPYAL	P,XT,loc32	Signed 32-bit multiply (upper half) and add previous P
QMPYL	ACC,XT,loc32	Signed 32 X 32-bit multiply (upper half)
QMPYL	P,XT,loc32	Signed 32 X 32-bit multiply (upper half)
QMPYSL	P,XT,loc32	Signed 32-bit multiply (upper half) and subtract previous P
QMPYUL	P,XT,loc32	Unsigned 32 X 32-bit multiply (upper half)
QMPYXUL	P,XT,loc32	Signed 32 X unsigned 32-bit multiply (upper half)

Direct Memory Operations

ADD	loc16,#16bitSigned	Add constant to specified location

Direct Memory Operations (nadaljevanje)

AND I	oc16,#16bitSigned	Bitwise AND
CMP	loc16,#16bitSigned	Compare
DEC	loc16	Decrement by 1
DMOV	loc16	Data move contents of 16-bit location
INC	loc16	Increment by 1
MOV	*(0:16bit),loc16	Move value
MOV	loc16,*(0:16bit)	Move value
MOV	loc16,#16bit	Save 16-bit constant
MOV	loc16,#0	Clear 16-bit location
MOVB	loc16,#8bit,COND	Store byte conditionally
OR	loc16,#16bit	Bitwise OR
TBIT	loc16,#bit	Test bit
TBIT	loc16,T	Test bit specified by T register
TCLR	loc16,#bit	Test and clear specified bit
TSET	loc16,#bit	Test and set specified bit
XOR	loc16,#16bit	Bitwise exclusive OR

IO Space Operations

IN	loc16,*(PA)	Input data from port
OUT	*(PA),loc16	Output data to port
UOUT	*(PA),loc16	Unprotected output data to I/O port

Program Space Operations

PREAD	loc16,*XAR7	Read from program memory
PWRITE	*XAR7,loc16	Write to program memory
XPREAD	loc16,*AL	C2xLP source-compatible program read
XPREAD	loc16,*(pma)	C2xLP source-compatible program read
XPWRITE	*AL,loc16	C2xLP source-compatible program write

Branch/Call/Return Operations

B	16bitOff,COND	Conditional branch
BANZ	16bitOff,ARn--	Branch if auxiliary register not equal to zero
BAR	16bOf,ARn,ARn,EQ/NEQ	Branch on auxiliary register comparison
BF	16bitOff,COND	Branch fast
FFC	XAR7,22bitAddr	Fast function call
IRET		Interrupt return
LB	22bitAddr	Long branch
LB	*XAR7	Long indirect branch
LC	22bitAddr	Long call immediate
LC	*XAR7	Long indirect call
LCR	22bitAddr	Long call using RPC
LCR	*XARn	Long indirect call using RPC
LOOPZ	loc16,#16bit	Loop while zero
LOOPNZ	loc16,#16bit	Loop while not zero
LRET		Long return
LRETE		Long return and enable interrupts
LRETR		Long return using RPC
RPT	#8bit/loc16	Repeat next instruction
SB	8bitOff,COND	Short conditional branch
SBF	8bitOff,EQ/NEQ/TC/NTC	Short fast conditional branch
XB	pma	C2XLP source-compatible branch
XB	pma,COND	C2XLP source-compatible conditional branch
XB	pma,*,ARPn	C2XLP source-compatible branch function call
XB	*AL	C2XLP source-compatible function call
XBANZ	pma,*ind{,ARPn}	C2XLP source-compatible branch if ARn is not zero
XCALL	pma	C2XLP source-compatible cal

Branch/Call/Return Operations (nadaljevanje)

XCALL	pma,COND	C2XLP source-compatible conditional call
XCALL	pma,*,ARPn	C2XLP source-compatible call with ARP modification
XCALL	*AL	C2XLP source-compatible indirect call
XRET		Alias for XRETC UNC
XRETC	COND	C2XLP source-compatible conditional return

Interrupt Register Operations

AND	IER,#16bit	Bitwise AND to disable specified CPU interrupts
AND	IFR,#16bit	Bitwise AND to clear pending CPU interrupts
IACK	#16bit	Interrupt acknowledge
INTR	INT1/../INT14	Emulate hardware interrupts
	NMI	
	EMUINT	
	DLOGINT	
	RTOSINT	
MOV	IER,loc16	Load the interrupt-enable register
MOV	loc16,IER	Store interrupt enable register
OR	IER,#16bit	Bitwise OR
OR	IFR,#16bit	Bitwise OR
TRAP	#0..31	Software trap

Status Register Operations (ST0, ST1)

CLRC	Mode	Clear status bits
CLRC	XF	Clear the XF status bit and output signal
CLRC	AMODE	Clear the AMODE bit
C28ADDR		Clear the AMODE status bit
CLRC	OBJMODE	Clear the OBJMODE bit
C27OBJ		Clear the OBJMODE bit
CLRC	M0M1MAP	Clear the M0M1MAP bit
C27MAP		Set the M0M1MAP bit

Status Register Operations (ST0, ST1) (nadaljevanje)

CLRC	OVC	Clear OVC bits
ZAP	OVC	Clear overflow counter
DINT		Disable maskable interrupts (set INTM bit)
EINT		Enable maskable interrupt (clear INTM bit)
MOV	PM,AX	Load product shift mode bits PM = AX(2:0)
MOV	OVC,loc16	Load the overflow counter
MOVU	OVC,loc16	Load overflow counter with unsigned value
MOV	loc16,OVC	Store the overflow counter
MOVU	loc16,OVC	Store the unsigned overflow counter
SETC	Mode	Set multiple status bits
SETC	XF	Set XF bit and output signal
SETC	M0M1MAP	Set M0M1MAP bit
C28MAP		Set the M0M1MAP bit
SETC	OBJMODE	Set OBJMODE bit
C28OBJ		Set the OBJMODE bit
SETC	AMODE	Set AMODE bit
LPADDR		Alias for SETC AMODE
SPM	PM	Set product shift mode bits

Miscellaneous Operations

ABORTI		Abort interrupt
ASP		Align stack pointer
EALLOW		Enable access to protected space
IDLE		Put processor in IDLE mode
NASP		Un-align stack pointer
NOP	{*ind}	No operation with optional indirect address modification
ZAPA		Zero accumulator P register and OVC
EDIS		Disable access to protected space

Miscellaneous Operations (nadaljevanje)

ESTOP0	Emulation Stop 0
ESTOP1	Emulation Stop 1