MPHL 8811

Bell Laboratories

subject: IS25 to BELLMAC-32A Assembly Language Instruction Mappings
Charge Case 12143-117
File Case 39382-900

date: October 1, 1982

from: Don Kretsch
MH 45421
3B-427 x7676
45421-821001.01MF

ABSTRACT

This document describes in detail the IS25 implementation for the BELLMAC-32A microprocessor.

IS25 is the assembly language instruction set specified for the 3B family of processors. Since the BELLMAC-32A is used in the 3B5 project it was necessary to implement IS25 on the BELLMAC-32A module. Since all the IS25 instructions were not available on the module the Software Generation System's assembler provides assembly language mappings to implement the missing IS25 instructions. These mappings are documented in this memorandum.

R. E. RICHTON
HL 1C-344

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by written approval of the director of the distributing organization.

Bell Laboratories

subject: IS25 to BELLMAC-32A Assembly Language Instruction Mappings
Charge Case 12143-117
File Case 39382-900

date: October 1, 1982

from: Don Kretsch
MH 45421
3B-427 x7676
45421-821001.01MF

MEMORANDUM FOR FILE

1. Introduction

This document describes in detail the IS25 [1] implementation for the BELLMAC*-32A microprocessor. IS25 is the assembly language instruction set specified for the 3B family of processors. It consists of a complete syntactic and semantic description of the assembly language as well as object level description of data values. Its purpose is to offer machine-independent "user-level" instructions that allow the development of portable assembly language programs. Since a machine requires many more instructions than those at the "user-level", IS25 is considered to be only a subset of the assembly language provided by a specific processor.

The IS25 specifications allow the processor to omit the direct implementation of some IS25 instructions provided that the omitted instructions can be functionally represented with other IS25 instructions that are directly implemented on the processor. This "instruction mapping" is the technique used by the BELLMAC-32A Software Generation System assembler to implement many IS25 instructions that are not directly available on the BELLMAC-32A chip.

Section 2 of this memo divides IS25 into four functional groups where each group represents a class of mappings that can occur. Section 3 gives the detailed expansions that occur in each of the groupings outlined in section 2. Floating point instructions are addressed as a separate issue in section 4.

2. Functional Groups

The IS25-to-BELLMAC-32A mappings can be divided into four groups. These groups are defined as follows:

---

1 Poplawski, D. A., "IS25 - The 3B Family Common Instruction Set - Release 2", 45181-820208.01MF.

* BELLMAC is a trademark of Western Electric.

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by written approval of the director of the distributing organization.

a. One-to-One

The IS25 instructions in this group are mapped directly into their BELLMAC-32A counterparts. These instructions represent the IS25 instructions that are directly executable on the BELLMAC-32A module with no instruction expansion or operand transformations.

b. Optimized One-to-One

A small subset of IS25 instructions can be mapped into a different, optimized form of their corresponding BELLMAC-32A instructions. These optimizations are not always applicable and can only occur under certain circumstances. If the required conditions are not met then these instructions are mapped directly into their BELLMAC-32A counterparts.

c. One-to-Many

This group represents IS25 instructions that have no BELLMAC-32A counterparts and therefore are not implemented on the module directly. Their function is accomplished through software mappings provided by the SGS assembler. In order for these expansions to be IS25 compatible two rules must be adhered to: the expansion must be done in terms of other IS25 instructions and addressing modes and the expansion cannot require register or memory allocation by the assembler.

d. Hardware Workarounds

These mappings exist due to current hardware problems. These represent unanticipated expansions that, hopefully, will not remain a permanent part of the SGS assembler.

The next section discusses each of the above expansions in detail. Each IS25 instruction is assigned to its appropriate section and, where applicable, its BELLMAC-32A mapping is provided.

3. Code Expansions

In the process of describing the various transformations the following semantics will be used.

Lower case mnemonics represent IS25 assembly language while upper case mnemonics represent BELLMAC-32A assembly language. Once an instruction has reached a BELLMAC-32A representation it will remain as such and will undergo no further transformations. Because of this the expansions given below represent an iterative process where the initial instruction may get expanded into a sequence of IS25 and BELLMAC-32A intermixed code. Any IS25 instructions generated will then be mapped into another sequence of code. This process continues until the final sequence contains only BELLMAC-32A instructions.

Square brackets indicate that one and only one of the elements contained within the brackets is to be selected.

a. One-to-One Mappings

The following IS25 instructions are mapped directly into their BELLMAC-32A counterparts. This means that only one BELLMAC-32A instruction is generated for each IS25 instruction in the following list.

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by
written approval of the director of the distributing organization.

One-to-One Mappings	
and[bhw]2	and[bhw]3
divw2	divw3
modw2	modw3
mulw2	mulw3
or[bhw]2	or[bhw]3
udivw2	udivw3
umodw2	umodw3
umulw2	umulw3
xor[bhw]2	xor[bhw]3
add[bhw]3	bit[bhw]
mcom[bhw]	mneg[hw]
movaw	movblw
pushaw	pushw
rsb	sub[bhw]3

b. Optimized One-to-One Mappings

The following IS25 dyadic instructions will be optimized into BELLMAC-32A monadic instructions when the first operand is a particular immediate constant.

IS25 Dyadic	Operand Constant	BELLMAC Monadic
mov[bhw]	0	CLR[BHW]
add[bhw]2	1	INC[BHW]
sub[bhw]2	1	DEC[BHW]
cmp[bhw]	0	TST[BHW]

If the first operand is not one of the above immediate values then the instruction is mapped directly into its BELLMAC-32 counterpart.

c. One-to-Many Mappings

The following list indicates the IS25 instructions that are implemented via instruction sequences. Similar instructions are grouped together for convienence and conciseness.

- Add, Compare and Jump Instructions

The mappings for the acjl, acjle, acjleu, and acjlu instructions are as follows:

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by written approval of the director of the distributing organization.

```
acX     index, incr, limit, dest

   where X = [jl, jlu, jle, jleu]

   is expanded into

addw2  incr, index
cmpw   index, limit
X      dest
```

- Add, Test, and Jump Non-Zero Instruction

  The atjnz[bhw] instructions are mapped as follows:

```
atjnzX  index, incr, dest

   where X = [bhw]

   is expanded into

addw2  incr, index
TSTX   *index
jnz    dest
```

- Compare Instruction

  The IS25 compare instructions take two arguments, like their BELLMAC-32A counterparts, but the order of the arguments must be reversed for the same result to occur. The mapping is as follows:

```
cmpX   opnd1, opnd2

   where X = [bhw]

   is expanded into

CMPX  opnd2, opnd1
```

  If, after the operands are reversed, the first argument (opnd2) is an immediate operand with a value of zero then an optimized form of this instruction is generated (see section 3b).

- Move Instructions

  The IS25 instruction set provides the user with the ability to convert a value from one data type to another in one instruction. These conversion instructions are not available on the module but are supplied by assembler mappings. In

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by
written approval of the director of the distributing organization.

understanding the expansions below it is helpful to keep in mind how the module does sign extension. Sign extension is determined by the type of the source operand. If the source operand is a signed byte or halfword the module will sign-extend; if it is an unsigned byte or halfword the module will zero-extend. Sign extension has no user-level effect on word operands. If the type of the source is not explicitly defined by the user then the following defaults apply: bytes are treated as unsigned quantities, halfwords and words are treated as signed quantities. The mappings are as follows:

Move Bit (Sign) Extended

```
movbbh  src, dst  →  MOVB  {sbyte}src, {shalf}dst

movbbw  src, dst  →  MOVB  {sbyte}src, {sword}dst

movbhw  src, dst  →  MOVH  src, {sword}dst
```

Move Zero Extended

```
movzbh  src, dst  →  MOVB  src, {uhalf}dst

movzbw  src, dst  →  MOVB  src, {uword}dst

movzhw  src, dst  →  MOVH  {uhalf}src, {uword}dst
```

Move Truncated

```
movthb  src, dst  →  MOVH  src, {sbyte}dst

movtwb  src, dst  →  MOVW  src, {sbyte}dst

movtwh  src, dst  →  MOVW  src, {shalf}dst
```

If the destination operand is a register then the above three truncate instructions are expanded as:

```
movthb  src, dst  →  ANDH3  &0xff, src, {ubyte}dst

movtwb  src, dst  →  ANDW3  &0xff, src, {ubyte}dst

movtwh  src, dst  →  MOVW  src, dst
                     MOVH  dst, dst
```

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by written approval of the director of the distributing organization.

- Push Instructions

IS25 offers instructions that allow the user to push a byte or halfword onto the stack as a 32-bit (word) value. These push instructions, pushb[bh] and pushz[bh], convert byte and halfword data types to word data types by specifying whether to sign extend or zero fill the resulting word value.

Each instruction is expanded into two BELLMAC-32A instructions. The first of these instructions is always the same while the second instruction is selected from a set of possible instructions depending on the operand. The mapping is:

```
push[zb]X       opnd

   where X = [bh]

   is expanded into

ADDW2           &4, %sp
```

The second instruction is selected from one of the following:

```
if opnd does not use %sp:
MOVX    {Z}opnd, {sword}-4(%sp)

if opnd = %sp:
SUBW3   &4, {Z}opnd, {sword}-4(%sp)

if opnd = expr(%sp):
MOVX    {Z}expr-4(%sp), {sword}-4(%sp)
```

where Z = sbyte (if X = byte),
uhalf (if X = half).

- Jump Instructions

The IS25 jump instructions are defined as follows:

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by written approval of the director of the distributing organization.

j[n]e	Jump (Not) Equal
j[n]z	Jump (Not) Zero
j[n]pos	Jump (Not) Positive
j[n]neg	Jump (Not) Negative
jg[e]	Jump Greater (or Equal)
jg[e]u	Jump Greater (or Equal) Unsigned
jl[e]	Jump Less (or Equal)
jl[e]u	Jump Less (or Equal) Unsigned
jmp	Jump Unconditional

If the destination of the jump instruction uses the external addressing mode (operand descriptor 0x7f) then it is possible to generate a branch relative to the value of the Program Counter (PC) register. If the destination of the jump instruction is from −128 to +127 from the current value of the PC, that is, it will fit into an 8-bit quantity, then the assembler will map the IS25 instruction into its BELLMAC-32A byte instruction counterpart. If the destination is outside the byte range but within the 16-bit range −32768 to +32767 of the current PC value then the IS25 instruction will be mapped into its halfword BELLMAC-32A counterpart. If the destination does not fit into either of the above quantities or if a PC relative branch cannot be generated because of the addressing mode then a two instruction sequence is generated. The first instruction is a complementary branch to the next instruction. The second instruction is an unconditional jump to the indicated destination.

The IS25 jump instructions, their BELLMAC-32A branch counterparts, and their complementary counterparts are given in the table below.

IS25 Jump	BELLMAC Branch	BELLMAC Comp. Branch
je jz	BE[BH]	BNE[BH]
jne jnz	BNE[BH]	BE[BH]
jg jpos	BG[BH]	BLE[BH]
jl jneg	BL[BH]	BGE[BH]
jle jnpos	BLE[BH]	BG[BH]
jge jnneg	BGE[BH]	BL[BH]
jgu	BGU[BH]	BLEU[BH]
jlu	BLU[BH]	BGEU[BH]
jleu	BLEU[BH]	BGU[BH]
jgeu	BGEU[BH]	BLU[BH]
jmp	BR[BH], JMP	---

- **Jump on Bit Status Instructions**

  The IS25 Jump on Bit Clear and Jump on Bit Set instructions, jbc and jbs respectively, are expanded as follows:

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by written approval of the director of the distributing organization.

```
jbX     pos, src, dest

   where X = [c, s]

   is expanded into

BITW  &(1<<pos), src
jY    dest

   where Y = z  (for jbc),
             nz (for jbs).
```

- Jump to Subroutine Instruction

  If the destination of the jump to subroutine instruction uses the external addressing mode (operand descriptor 0x7f) then it is possible to generate a branch relative to the value of the Program Counter (PC) register. If the destination is from −128 to +127 from the current value of the PC, that is, it will fit into an 8-bit quantity, then the assembler will map the IS25 instruction into the BELLMAC-32A BSBB instruction. If the destination is outside the byte range but within the 16-bit range −32768 to +32767 of the current PC value then the IS25 instruction will be mapped into its halfword BELLMAC-32A counterpart, BSBH. If the destination does not fit into either of the above quantities or if a PC relative branch cannot be generated because of the addressing mode then the BELLMAC-32A JSB instruction is generated.

- Shift Instructions

  IS25 defines left and right shift operations on both two and three operand instructions. In IS25 the shift count is defined to be a byte operand. The module implements these shifts on the data types byte, halfword, and word, however it only allows the triadic version. The IS25 dyadic shifts, alsw2, arsw2, llsw2, and lrsw2 are therefore mapped into their triadic counterparts with the third operand being identical to the second operand.

  The assembler attempts to use the byte mnemonics whenever possible. If the shift count operand is a register or an immediate value then no expansion of the shift instruction occurs and it is directly mapped into its BELLMAC-32A shift instruction. Otherwise the mapping is as follows:

Arithmetic Shifts

```
alsw3   count, src, dest

   is expanded into

ALSW3   {sbyte}count, src, dest
```

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by written approval of the director of the distributing organization.

arsw3 count, src, dest

is expanded into

ARSB3 count, src, dest

Logical Shifts

llsw3 count, src, dest

is expanded into

LLSB3 count, src, dest

lrsw3 count, src, dest

is expanded into

LRSW3 {sbyte}count, src, dest

- Bit Field Instructions

Both IS25 and the BELLMAC-32A define bit field manipulation instructions. These instructions, insert bit field and extract bit field, perform identical operations, but the order and values of the specific operands differ slightly. The assembler maps them as follows:

Insert Field

insv src, offset, width, dest

is expanded into

INSFW width-1, offset, src, dest

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by written approval of the director of the distributing organization.

Extract Field

```
extzv   offset, width, src, dest
```

is expanded into

```
EXTFW   width-1, offset, src, dest
```

- Move Block Instructions

The IS25 move block word instruction is directly implemented on the module, but the byte and halfword versions of this instruction are not. Therefore, the assembler maps these instructions as follows:

movblb

is expanded into

```
TSTW    %r2
BRB     0xb
MOVB    0(%r0), 0(%r1)
INCW    %r1
INCW    %r0
DECW    %r2
BNEB    -9
```

movblh

is expanded into

```
TSTW    %r2
BRB     0xf
MOVH    0(%r0), 0(%r1)
ADDW2   &2, %r1
ADDW2   &2, %r0
DECW    %r2
BNEB    -13
```

- Call Instruction

The IS25 call instruction takes two arguments: the number of parameters passed and the name of the function to be called. This maps into the BELLMAC-32A CALL instruction as:

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by written approval of the director of the distributing organization.

```
call    num, dest
```

is expanded into

```
CALL  -(4*num)(%sp), dest
```

- Save / Return Instructions

The IS25 save instruction takes as its operand the number of registers to be saved. The BELLMAC-32A SAVE instruction takes as its operand a register addressing mode where the number of registers saved is the register number specified by the save operand subtracted from nine. For example, a BELLMAC-32A SAVE operand of %r3 will save (9 - 3) = 6 registers.

The assembler supplies the following mapping:

```
save    num
```

is expanded into

```
SAVE  %rn
```

where n = (9 - num).

The IS25 return instruction takes as its argument the number of register to be restored when control is passed back to the calling program. This value should be the same as indicated in the function's save instruction. The assembler supplies the following mapping:

```
ret     num
```

is expanded into

```
RESTORE     %rn
RET
```

where n = (9 - num)

d. Hardware Workarounds

This section documents mappings that the assembler must perform in order to eliminate certain instruction sequences due to hardware deficiencies. The mappings listed below affect both IS25 and BELLMAC-32A instructions and are included here for completeness and documentation purposes.

- Triadic Multiply

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by written approval of the director of the distributing organization.

Due to a timing problem on mask 1 of the BELLMAC-32A DBO module the IS25 triadic instructions mulw3, divw3, modw3, their unsigned versions (umulw3, udivw3, umodw3) and the BELLMAC-32A MUL[BHW]3, DIV[BHW]3, MOD[BHW]3 instructions must be mapped into their BELLMAC-32A dyadic counterparts. The mapping is as follows:

"instr"X3 arg1, arg2, arg3

where X = [BHWw]

is expanded into

```
pushYX      arg1
MOVX        marg2, marg3
"instr"X2   {Z}expr(%sp), marg3
SUBW2       &4, %sp
TSTX        arg3
```

if arg2 is the stack pointer (%sp) then the second instruction above is replaced by:

```
SUBW3       &4, %sp, marg3
```

where: Y = z (zero extend; if X = byte),
b (bit extend; if X = half).

Z = ubyte (if X = byte),
shalf (if X = half).

expr = -1 (bytes), -2 (halfs),
or -4 (words).

marg[2|3] indicates arg[2|3] was adjusted if it referenced the stack.

This transformation is necessary for the first mask of the DBO module. When the DBO mask 2 module is verified to correctly handle these triadic instructions this mapping should be removed from the SGS assembler.

- Memory Writes

It is possible for the Program Counter (PC) to be incorrectly updated when an instruction is performing a memory write operation and an interrupt occurs. The SGS solution is to generate a one byte no-operation (NOP) after each instruction using a memory-based operand as its destination. The following instructions are exempt from this fix: CALL, SAVE, PUSHW, PUSHAW, and the IS25 call, save, pushw, and pushaw. These instructions need not be (and are not) modified. All other IS25 or BELLMAC-32A instructions are mapped as follows:

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by written approval of the director of the distributing organization.

"instruction" opnd

where opnd represents any memory write addressing mode.

is expanded into

"instruction" opnd
NOP

This expansion will remain in the SGS assembler indefinitely since there are no current plans to fix the chip.

- TSTW and Interrupts

If the chip responds to an interrupt after a specific sequence of instructions then the last instruction in the sequence may not set the condition code bits in the PSW register.

The problem appears in a two instruction sequence where the first instruction must fit either one of the following criteria:

1. Instruction does a store to a register and the last source operand comes from memory. This does not include any discontinuous instructions like SWAP or RESTORE or instructions without destinations (e.g. BIT, TST, CMP).
2. Instruction does a store to a register and is a multiple cycle ALU operation (e.g. MUL, DIV, MOD, INSF, MNEG).

The second instruction in the sequence must be one of the following: CLRW, DECW, INCW, MCOMW, MOVW or TSTW with all register operands.

The SGS fix for this problem is the insertion of a one byte NOP between the two instructions in the sequence. In an attempt to avoid the occurrence of this problem the IS25 instructions which can generate the second instruction in the sequence, namely the optimized one-to-one instructions, are also affected. The optimizations outlined in section 3b will not be done if the destination operand is a register AND the previous instruction is the first instruction of the above sequence.

This expansion will remain in the SGS assembler indefinitely since there are no current plans to fix the problem in hardware.

4. Floating Point

The SGS compiler and assembler accept floating-point code written in C and IS25 assembly language respectively. Although floating point is not available directly on the BELLMAC-32A module it can be implemented via a standard illegal opcode UNIX signaling mechanism. When an illegal opcode signal occurs a special floating point library is called to handle the floating point instruction. If it is not a floating point instruction then the signal is left unchanged and the operating system handles it in an appropriate

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by
written approval of the director of the distributing organization.

manner.

The SGS assumes such a mechanism exists (as it does in the 3B5 project) and generates floating point opcodes that the module considers illegal. These floating point opcodes are listed here just for IS25 completeness. No additional expansions or instruction mappings occur.

Single Precision Dyadic	
Mnemonic	Opcode
fadds2	0x31
fsubs2	0x41
fmuls2	0x51
fdivs2	0x61
movhs	0x71
movws	0x81
movss	0x91
movsd	0xa1
movsh	0xb1
movtsh	0xc1
movsw	0xd1
movtsw	0xe1
fcmps	0xf1

Double Precision Dyadic	
Mnemonic	Opcode
faddd2	0x35
fsubd2	0x45
fmuld2	0x55
fdivd2	0x65
movhd	0x75
movwd	0x85
movds	0x95
movdd	0xa5
movdh	0xb5
movtdh	0xc5
movdw	0xd5
movtdw	0xe5
fcmpd	0xf5

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by written approval of the director of the distributing organization.

Single/Double Triadic	
Mnemonic	Opcode
fadds3	0x39
fsubs3	0x49
fmuls3	0x59
fdivs3	0x69
faddd3	0x79
fsubd3	0x89
fmuld3	0x99
fdivd3	0xa9

5. Summary

This memorandum documents the IS25 implementation used by the SGS assembler for the BELLMAC-32A microprocessor. There are four classes of instruction set mappings. These classes include: the degenerate case of one-to-one mappings, that is, no change to the assembly language instruction; an optimized form of a one-to-one mapping; the one-to-many mapping that allows the implementation of IS25 instructions that do not exist on the module; and the temporary software patches for hardware problems. Each IS25 assembly instruction was assigned to a class and its mapping was defined.

An attempt was made to be very explicit with the one-to-many mappings since they are sometimes very obscure and probably one of the least understood parts of the SGS assembler's job.

Don Kretsch

MH-45421-DJKJR-djk

Don Kretsch

Copy to
All Supervision 4513, 45144, 45145, 45146, 4542
All Members Language Groups in 45421
E. L. Hedrick
M. E. Isenman
R. A. Kasten
L. L. Kopecky
M. J. Killian
P. M. Lu
D. A. Poplawski
D. A. Ramsey
R. E. Richton
B. F. Snow
D. A. Tafuri
D. E. Thompson-Matthews

BELL LABORATORIES PROPRIETARY
Not for use or disclosure outside Bell Laboratories except by written approval of the director of the distributing organization.