Advanced
Micro
Computers
A subsidiary of
Advanced Micro Devices

TRANZ Translator

User's Manual

$5.00

	REVISION RECORD
REVISION	**DESCRIPTION**
01	Preliminary Issue
(4/27/79)	
A	Manual Released
(11/16/79)	
B	Manual updated to correct documentation errors and add Z80 Translation Summary.
(3/19/80)	
C	Manual reprinted
(5/28/80)	
Publication No. 00680118	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning
this manual to:

ADVANCED MICRO COMPUTERS
Publications Department
3340 Scott Boulevard
Santa Clara, CA 95051

PREFACE

TABLE OF CONTENTS

CHAPTER 1
TRANZ PRODUCT DEFINITION

1-1. INTRODUCTION

The AMC translator utility TRANZ is a programming tool designed to aid in upgrading standard 8080/8085/Z80 assembler language source code to the AmZ8000 instruction set and format. TRANZ executes in an AmSYS 8/8 environment in 64K bytes of memory, translating source statements into functionally identical AmZ8000 statements. Registers specified in source-code operands are automatically mapped to the functionally equivalent AmZ8000 registers.

Where functional equivalencies do not exist, the non-translatable source statements are passed unchanged through the translator and are flagged for user action. (Note that TRANZ does not include a facility for interrupt translation.) In some cases (e.g., where exact memory-ordering or flag-bit setting is required), one line of source code can result in multiple lines of object code from the translator.

The output of TRANZ is compatible with the AmZ8000 assembler MACRO8000 and can be used as the source-code input to that utility. (Refer to the MACRO8000 Assembler User's Manual for usage details.)

The translation process is user-controlled through a series of commands to specify: translation qualities for byte ordering and flag-bit setting, ordering and testing; input format; output (object file) content; screen display content; and the selective translation, deletion, replacement, or copying of source code.

Translator actions may be repeated until the end of the source file, for a specified number of source lines, until a key word is reached, or until a translation problem of a specified severity occurs.

Translator commands can be embedded in the source code for execution in a batch mode, or can be specified interactively by the user after invoking TRANZ. Translation parameters are entered in response to screen prompts to specify translator actions (translate, delete, replace, copy list options, execute stored command, or quit) and translation mode settings (varying the exactness of the translation, display and output selection, input column selection, or set the stored command). Diagnostic messages flag user-command errors, source errors, and notes, warnings and errors in translation.

1-2. TRANSLATOR MODES

TRANZ can be commanded to execute either interactively or in batch mode.

1-3. INTERACTIVE MODE

In Interactive mode, translator commands and parameters are specified through user-entered responses to screen prompts for a command. Commands are not executed until the entire command-parameter set is correctly entered and terminated with a command delimiter (NEW LINE keystroke, blank, or non-alphanumeric characters). Note that in Interactive mode, commands also can be embedded in the source code.

1-4. BATCH MODE

In batch mode, translation commands (if any) are placed in the source code (in line) prior to invoking TRANZ. The in-line commands are then executed as they are encountered in the code. Batch mode setting is essentially the equivalent of interactive mode with TRANSLATE UNTIL END OF FILE action specified.

1-5. IN-LINE COMMAND SPECIFICATION

Any allowable command may be placed in the source code by preceding the command with an exclamation point in column 1. As an example, the source-code line entry

```
!MTO,
```

commands that only the translated object lines are to be displayed at the user's terminal beginning at the point at which the embedded command is encountered. The command is evaluated as

M	identifies the in-line entry as change to mode-setting commands
T	specifies terminal display selection
O	specifies that all translated source-program lines are to be displayed as they occur
,	terminates the command string.

In-line commands must be entered as shown above; blanks may not be used to separate the members of a command set, or between the exclamation point, the command set and the terminating comma. (Blanks are intrepreted by TRANZ as terminators)

1-6. TRANSLATION QUALITIES

The user has the option of specifying the precision of the translation; that is, the exact simulation of flag settings, flag-bit ordering and testing, and byte ordering. By using the "Translate Until..." option, translation qualities may be altered during the course of the translate operation. With most memory-transfer and arithmetic instructions however, the translated differences in flags and memory-ordering do not effect execution of a program. For this reason, and because an exact simulation requires generation of multiple instructions (on the order of 5) for each source instruction, it is recommended that all translation qualities be set to Don't Care condition when possible. In particular, setting the flag-bit ordering quality (F) to Care mode in PUSH PSW and POP PSW instructions requires the addition of two subroutines to the output code. (The subroutines are contained in Appendix B.)

1-7. REVERSE BYTE ORDERING, MEMORY (B)

The B quality set to Care condition specifies adjustment for the ordering of bytes as they appear in memory. The order in which bytes are stored in memory varies between the source CPUs and the AmZ8000. Further, the Z80 assembler reverses the order in which the high and low bytes of 16-bit word declaration values are stored, while the 8080 and 8085 reverse the bytes in both word-declaration values and strings. The effect of the source CPU byte ordering and of setting the TRANZ translation B qualilty to Care and Don't Care conditions are shown in table 1-1. The right-hand columns list the order in which DW 'string', DW VALUE, and DB 'string' are (1) stored in memory, (2) referenced by word, and (3) referenced by byte. Within each of the three categories, the byte order is defined for each source CPU, and for the translated AmZ8000 code with the B quality setting as both Care and Don't Care.

Adjustment is accomplished by a combination of reversing the byte order of some declarations (i.e., DW) and by combining word size load and store operations with byte exchanges. If data defined in a DW instruction is referenced by bytes, or if data defined in a DB instruction is referenced by words, the B quality Care condition is required.

However, setting this quality to Care can cause a significant degradation of 16-bit address and data processing. The B quality default setting is to Don't Care condition.

1-8. REVERSE BYTE ORDERING, STACK (K)

The K translation quality specifies adjustment for the reverse byte ordering of words as they appear on the stack. This is necessary since the user could reference the stack as if it were normal memory. Setting the K translation quality will significantly impact stack references, including subroutine linkage. The K quality default setting is to Don't Care condition.

TABLE 1-1. BYTE ORDERING FOR STORAGE, REFERENCE

	Source CPU Directives					
	DW 'STRING'		DW Value		DB 'STRING'	
	8080/85	Z80	8080/85	Z80	8080/85	Z80
(1) Storage:						
Source CPU	HL	LH	HL	HL	LH	LH
AmZ8000						
B = Don't Care	LH	HL	LH	LH	LH	LH
B = Care	HL	LH	HL	HL	LH	LH
(2) Word Reference:						
Source CPU	CR	CR	CR	CR	CR	CR
AmZ8000						
B = Don't Care	C	C	C	C	*	*
B = Care	CR	CR	CR	CR	CR	CR
(3) Byte Reference:						
Source CPU	C	C	C	C	C	C
AmZ8000						
B = Don't Care	*	*	*	*	C	C
B = Care	C	C	C	C	C	C

HL = High-low byte storage
LH = Low-high byte storage
C = Compatible byte order reference
CR = Compatible reference, byte reversed
* = Incompatible (byte-reversed) value reference

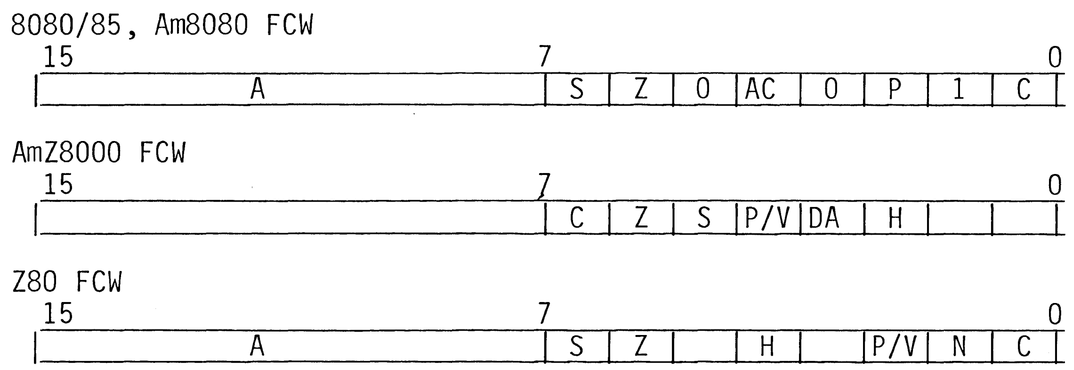
1-9. FLAG CONTROL WORD FLAG-BIT ORDERING (F)

Setting the F translation quality to Care condition requires that the flag bits in the Flag Control word (FCW) be ordered such that source-code references to the flag word are exactly simulated in the output code. The Care/Don't Care status of the F quality must be identical for the PUSH/POP and CALL/RETurn instructions. The FCW formats for the Intel 8080/85, Z80, Am8080 and AmZ8000 CPUs are shown in figure 1-1. The F quality default is to Don't Care mode.

1-10. DEFAULT FCW FLAG-BIT TRANSLATION QUALITIES

The Parity (P,P/V), Auxiliary Carry (AC,H) and Decimal Adjust (DA,N) qualities default to Don't Care condition; the Carry (C), Zero (Z), and Sign (S) qualities all default to Care condition. As previously noted, it is recommended that the translation qualities be set to Care condition only when required. In particular, P or P/V should remain in Don't Care condition wherever possible, since setting this quality to Care will increase the amount of code generated for many 8080/8085 instructions.

The D quality set to Care condition specifies adjustment for the effects of using an 8080/8085 DAA instruction in a subtract operation. (This instruction sets the DA (N) flag for decimal adjust after increment and add operations, but not after subtract or decrement operations.) It is recommended that the user include the necessary decimal-adjust code in the source rather than setting the D quality to Care condition since it can seriously degrade BCD arithmetic. The D quality default setting is to Don't Care condition.



S	- Sign flag
Z	- Zero flag
AC	- Auxiliary (Half) Carry (Same as H)
P	- Parity flag (Same as P/V)
C	- Carry flag
P/V	- Parity/Overflow flag
DA	- Decimal Adjust flag
H	- Half Carry flag
N	- Add/Subtract flag

Figure 1-1. FCW Formats

1-11. TRANSLATION QUALITY SELECTION SUMMARY

Table 1-2 summarizes the translation quality options and the default settings for each quality.

TABLE 1-2. TRANSLATION QUALITY OPTIONS

Quality	Default Setting*
A (AC,H) - Auxiliary (Half) Carry flag bit	D
C - Carry flag bit	C
P (P,P/V) - Parity (Parity/Overflow) flag bit	D
S - Sign flag bit	C
Z - Zero flag bit	C
D (DA,N) - Decimal adjust	D
B - Byte ordering, memory	D
K - Byte ordering, stack	D
F - FCW Flag-bit ordering	D
* D = Don't Care C = Care	

1-12. INSTRUCTION TRANSLATION

This subsection defines the TRANZ capabilities for translating instruction operands. The specific translations performed for each instruction and the AmZ8000 equivalents of the 8080/8085 and Z80 assembly language pseudo-operators are defined in the appendices to this manual.

1-13. ADDRESS TRANSLATION

TRANZ corrects the constant values in address relative operands (e.g. JMP $+4, JNZ label+4) for the increased size of the AmZ8000 code produced; however, absolute address operands and RST and PCHL instructions are flagged to the user with a warning. Additionally, operands and pseudo-operands (such as the Intel % operator) are ignored and flagged to the user when there is no equivalent translation.

1-14. REGISTER-REFERENCE TRANSLATION

Source code register-reference operands are translated to the comparable AmZ8000 registers, as defined in table 1-3.

TABLE 1-3. REGISTER MAPPING

Source Registers			AmZ8000 Register	
	SP		R15	
8080/8085 and Z80 Register Set	F	A	RH0	RL0
	B	C	RH1	RL1
	D	E	RH2	RL2
	H	L	RH3	RL3
Z80 Alternate Register Set	FA′		R8	
	BC′		R9	
	DE′		R10	
	HL′		R11	
Z80 Index Registers	IX		R6	
	IY		R7	
Translator Registers	Working byte register		RL4	
	Working word register		R5	

1-15. IDENTIFIER TRANSLATION

Identifiers are not modified by TRANZ except that ignorable characters are removed. The conversion of source code identifiers to valid TRANZ identifiers must be performed by the user prior to translation, as follows:

- The user must translate the special characters ? , # and $ to legal characters. These include A...Z (except for Z80 assembler code, equivalent to a...z), 0...9, and @.

- For Z80 source code, the user must ensure that all lower case variables used are equivalent to upper case variables, since in Z80 assembly language A...Z is not the same as a...z.

- Ambiguous label and EQU identifiers are found by the object (AmZ8000) assembler; ambiguous SET identifiers are not found and may result in incorrect assembly.

- Reserved words (i.e., AmZ8000 opcodes or pseudo-opcodes) used as identifiers in the source code are flagged to the user as warnings.

- All identifiers not declared prior to reference are treated by TRANZ as labels.

CHAPTER 2
USING TRANZ

2-1. INTRODUCTION

Execution parameters for TRANZ are specified through user responses to screen prompts. Translator command prompts are presented in two classes: actions and mode settings. Mode settings define the required exactness of the translation; action commands specify operations such as copy, delete, insert source statements, etc.

Unless noted in the descriptions, each command line must be terminated with a delimiter (NEW LINE, blank, or a non-alphanumeric character) before the command can be accepted by TRANZ. Parameters can be commanded in a single input line with the system call or they may be entered singly in response to each prompt.

When invoked, TRANZ fetches the translation-table file for the user-specified assembly language: filename TRANZ of type .I, .5, or .Z. A temporary file, TRANZ.$$$, is created as a work file for the translation process. The temporary work file, which is approximately twice the size of the output file to accommodate intermediate translate operations, is deleted after the translation is completed. The output of TRANZ is an ASCII file of default type .ZSC, the default input filetype to MACRO8000. If an output filename is not specified, the source filename is used. The user also can specify a zero output, in which case an output file is not generated.

2-2. CONVENTIONS USED

Within this chapter, the following conventions are used in defining commands and system prompts/responses:

___	Underlines indicate data to be supplied by the user.
[ ]	Square brackets indicate that the enclosed parameter or partial parameter is optional.
lowercase	Lowercase letters indicate variables for which values are to be supplied by the user or by TRANZ as output.
UPPERCASE	Uppercase letters indicate prompts displayed by TRANZ to request a user command, or responses to a user-entered command. Entries must be input exactly as shown.

2-3. INVOKING TRANZ

TRANZ can be run in either of two modes: Interactive or Batch. In Interactive mode, commands and execution parameters are specified through responses to TRANZ prompts, although commands and parameters also may be embedded in the source code. (Refer to chapter 1 for a description of the in-line command format.) In Batch mode, all commands and parameters must be in-line in the source code. The translator is invoked by entering

```
XPAS TRANZ
```

TRANZ responds with

```
8080, 8085 AND Z80 to Z8000 TRANSLATOR
TRANZ n.n mm/dd/yy
```

where n.n is the version level and mm/dd/yy is the version date. This is followed by the prompts

```
ENTER SOURCE (DEFAULT TYPE IS .ASM): sfile[.typ]

ENTER DESTINATION (DEFAULT sfile.ZSC): [dfile[.typ]]
```

where

sfile[.typ] is the disk file containing the source assembly language.

[dfile[.typ]] is the user-assigned disk file into which the translated code is to be placed. If omitted, the destination filename defaults to that of the source file, of default type .ZSC, the MACRO8000 default input file type. (Note that if the destination file is assigned as Ø, no output file is produced.)

The next message

```
ENTER SOURCE ASSEMBLY LANGUAGE (I,5,Z): __
```

prompts for the one of the following codes to specify the source assembly language:

```
I = Intel 8080
5 = Intel 8085
Z = Z80
```

Finally, the message

```
ENTER OPERATION (I,B): __
```

prompts for the operating mode, either Interactive or Batch. Selection of Batch operating mode is the equivalent of an action command to translate until the end of the file. Code translation begins after the

B command delimiter is encountered. Selection of Interactive operating mode causes TRANZ to prompt with the action command line

```
  ACTION (T,C,R,D,X,M,L,Q):_
```

requesting user entry of the first translator action command.

Since the prompts to invoke TRANZ are order-dependent, the call also can be given in a single line of the form

```
      XPAS TRANZ sfile[.typ],[dfile[.typ]],language,operation
```

As an example, the call

```
      XPAS TRANZ SFILE,,Z,B.
```

specifies source filename SFILE of default type .ASM, destination filename (default) SFILE.ZSC, source assembly language Z80, batch operation. In this command-line format, a comma must be inserted to indicate that the destination parameter has been omitted. Note the use of a period as the call delimiter.

After the source assembly language is specified, TRANZ loads the appropriate translation table and translates the source file to AmZ8000 assembly language statements.

2-4. INTERACTIVE TRANSLATION COMMANDS

NOTE

If a command response is incorrectly entered, Control-X or Control-U can be used to cancel the current command line. The command prompt will be immediately redisplayed and TRANZ will halt and wait for the response entry.

When translating interactively, the user selects translation conditions through responses to prompts issued by TRANZ in two major classes: Action and Mode-Setting. The prompt line for each class includes codes assigned to each translation parameter command in that class. The user enters the desired code on the prompt line, and TRANZ verifies the selection by completing the code word and adding any subcodes within the command. A selected subcode is verified in the same manner. As an example, given the Action command line

```
 ACTION (T,C,R,D,X,M,L,Q):
```

a user selection of code T would result in the line

```
 ACTION (T,C,R,D,X,M,L,Q): TRANSLATE UNTIL (E,S,WD,#):
```

selecting subcode E results in line completion as

```
 ACTION (T,C,R,D,X,M,L,Q): TRANSLATE UNTIL (E,S,WD,#): END OF FILE
```

As a result of this sequence, the entire file will be translated, with all specified mode settings.

By entering a command delimiter (blank, NEW LINE, or non-alphanumeric keystroke) when the ACTION or MODE SETTING line is displayed, the user can command a display that lists all options in that class. (The ACTION/MODE SETTING command line remains active on the screen during the display.) The Action and Mode Setting class options are shown in figures 2-1 and 2-2. Note that each command class includes an option that allows the user to switch between classes.

A	ACTION CLASS COMMANDS
T	TRANSLATE SOURCE
C	COPY SOURCE
R	REPLACE SOURCE
D	DELETE SOURCE
X	EXECUTE STORE (stored command 40 bytes)
M	MODE SETTING CLASS COMMANDS
L	LIST TRANSLATION QUALITIES
Q	QUIT

Figure 2-1. Action Class Options

M	MODE SETTING CLASS COMMANDS
L	LIST TRANSLATION QUALITIES
C	CARE
D	DON'T CARE
T	TERMINAL DISPLAY SELECTION
O	OBJECT SELECTION
I	INPUT COLUMN SELECTION
S	STORE COMMAND (stored comand 40 bytes)
A	ACTION CLASS COMMANDS

Figure 2-2. Mode-Setting Class Options

2-5. ACTION COMMANDS

Action commands direct the translation, deletion, replacement and insertion of source code into the selected terminal and object file output. After the selection of Interactive operation, TRANZ displays the Action command line prompt

```
ACTION (T,C,R,D,X,M,L,Q): _
```

and waits for a response. After each action command is entered and processed as defined by the mode-setting parameters, and all in-line commands encountered have been processed, the ACTION prompt is redisplayed and TRANZ waits for further user input.

After execution of the conditional action commands T (Translate until x), C (Copy until x), R (Replace until x) and D (Delete until x), the current status is displayed as shown in figure 2-3, and TRANZ prompts for entry of another action command. In the status display, e is the error severity code letter and nnn represents line numbers and the current program counter location.

```
ERROR SEVERITY          =     e
SOURCE PC               =     nnn
OBJECT PC               =     nnn
CURRENT SOURCE LINE     =     nnn
(Current source line 80 bytes)
```

Figure 2-3. Current Status Display

NOTE
All ACTION commands must be terminated with a delimiter (NEW LINE, blank, or non-alphanumeric character) before they will be accepted by TRANZ for processing.

2-6. ACTION: Translate Until

Translate Until specifies the condition under which translation of the source assembler language statements is to terminate.

```
ACTION (T,C,R,D,X,M,L,H): TRANSLATE UNTIL (E,S,'WD',#): __
```

A command delimiter results in translation of one line of input file source code, and processing of the specified output; i.e., terminal and object output as defined in the mode setting.

```
...TRANSLATE UNTIL (E,S,'WD',#): END OF FILE
```

An E response results in translation of the entire file, using selected mode-setting options.

```
...TRANSLATE UNTIL (E,S,'WD',#): SEVERITY (N,W,Z)
```

An S response results in translation of the file until a problem of the selected severity is encountered, where

N = translate until a source line is encountered that cannot be fully translated (possibly due to translation quality selections), or until translator address recalculation must be done. Note that entering a command delimiter at this point has the same effect as an N response.

W = translate until a source line is encountered for which a fix routine is not available.

Z = translate until a source line is encountered that cannot be translated, or until a command error is encountered.

```
...TRANSLATE UNTIL (E,S,'WD',#): 'string'
```

Selection of 'WD' (keyword) is in the form of a string of up to sixteen characters, bounded with single quotes. (Note that the code 'WD' need not be entered.) When the specified string is encountered in the source line, TRANZ halts and prompts with the ACTION display for user intervention.

```
...TRANSLATE UNTIL (E,S,'WD',#): n
```

Selection of # is in the form of an integer value n. (Note that the code # need not be entered.) TRANZ processes n source lines, halts and prompts with the ACTION display for user intervention.

2-7. ACTION: Copy Until

Copy Until allows specification of the condition under which the copy of source lines into untranslated object-file lines is to stop.

```
ACTION (T,C,R,D,X,M,L,H): COPY UNTIL (E,'WD',#): __
```

Entering a command delimiter results in the output (copy) of one line of untranslated source code as defined by the mode-setting options for the selected output.

```
...COPY UNTIL (E,'WD,'#): END OF FILE
```

An E response results in a copy of the entire file as defined by the mode-setting options.

```
...COPY UNTIL (E,'WD',#): 'string'
```

Selection of 'WD' (keyword) is in the form of a string of up to 16 characters, bounded with single quotes. (Note that the code 'WD' is not required.) When the specified string is encountered in a source line, the copy operation halts and TRANZ prompts with the ACTION display for user intervention.

```
...COPY UNTIL (E,'WD',#): n
```

Selection of # is in the form of an integer value n. (Note that the code # is not required.) TRANZ copies the next n source lines, halts and prompts with the ACTION display for user intervention.

2-8. ACTION: Replace Until

Replace Until specifies the termination point for replacement of source code with user-entered code.

```
ACTION (T,C,R,D,X,M,L,H): REPLACE UNTIL (E,'WD',#): _
```

Entering a command delimiter causes TRANZ to prompt with

```
REPLACE WITH (TERMINATED BY //):
```

and wait for replacement code, terminated with a double slash, to be entered at the terminal. The code is then processed as defined by the selected mode-setting parameters. The command delimiter specifies a single replacement line, by default.

```
...REPLACE UNTIL (E,'WD',#): END OF FILE
```

An E response results in replacement of the remaining source file with (untranslated) code entered at the console. The code is then processed as defined by the mode-setting parameters.

```
...REPLACE UNTIL (E,'WD',#): 'string'
```

Selection of 'WD' (keyword) is in the form of a string of up to 16 characters, bounded with single quotes. (Note that the code 'WD' is not required.) When the specified string is encountered in a source line, the replace operation halts and TRANZ prompts with the ACTION display for user intervention.

```
  ...REPLACE UNTIL (E,'WD',#): n
```

Selection of # is in the form of an integer n. (Note that the code # is not required.) TRANZ replaces the next n lines of source code with the user-entered code, and processes the code as defined by the mode-setting parameters for the terminal and object output.

2-9. ACTION: Delete Until

The Delete option does not delete the specified source-code lines; rather, TRANZ ignores these lines during the translation operation. Thus, Deleted lines remain in the source code but are not copied as part of the object code.

Delete Until specifies the point at which the deletion of source assembly language code is to terminate, with no untranslated assembly language object code produced.

```
  ACTION (T,C,D,R,X,M,L,H): DELETE UNTIL (E,'WD',#):_
```

Entering a command delimiter causes TRANZ to delete (skip) the current source line.

```
  ...DELETE UNTIL (E,'WD',#): END OF FILE
```

An E response results in deleting (skipping) the file from the current source line through the end of the file.

```
  ...DELETE UNTIL (E,'WD',#): 'string'
```

Selection of 'WD' (keyword) is the form of a string of up to 16 characters, bounded with single quotes. (Note that the code 'WD' is not required as part of the entry.) When the specified string is encountered at the beginning of a source line, the delete (skip) operation halts and TRANZ prompts with the ACTION display for user intervention.

```
  ...DELETE UNTIL (E,'WD',#): n
```

Selection of # is in the form of an integer n. (Note that the code # is not required.) TRANZ will skip the next n lines of source code.

2-10. ACTION: nX (Execute)

The user can create a stored command, consisting of up to 40 bytes of action, class-switching, and mode-setting commands, by specifying the S mode-setting option. (Refer to paragraph 2-21 for a description of the String mode-setting command.) The action command nX causes the stored command to execute n times. When n is not specified, the default value is 1. Note that both the S and X commands can be nested.

2-11. ACTION: Mode-Setting

Entering an M response causes TRANZ to switch to mode-setting class and prompt for a command. (This command is executed when it is received; a delimiter may not be used to terminate the MODE command.)

```
ACTION (T,C,R,D,X,M,L,H): MODE SETTING

MODE (L,C,D,T,O,I,S,A):
```

TRANZ continues to prompt for mode-setting class commands until an A (Action) command is issued.

2-12. ACTION: List

Entering an L response causes TRANZ to display the current translation-quality settings and register mapping, as shown in figures 2-4 and 2-5.

```
m   SOURCE MACHINE          a    ASSEMBLY LANGUAGE
q   C       CARRY FLAG
q   Z       ZERO FLAG
q   S       SIGN FLAG
q   P       PARITY FLAG
q   A       AUXILIARY CARRY FLAG
q   F       FLAG BIT ORDER
q   D       DECIMAL ADJUST SUBTRACT
q   B       BYTE ORDER IN MEMORY
q   K       BYTE ORDER ON STACK
```

Figure 2-4. Translation Quality Status

Source		Object	
SP		R15	
F	A	RH0	RL0
B	C	RH1	RL1
D	E	RH2	RL2
H	L	RH3	RL3
Working Byte Register		RL4	
Working Word Register		R5	

Figure 2-5A. 8080/8085 Register Mapping

Source		Object	
SP		R15	
F	A	RH0	RL0
B	C	RH1	RL1
D	E	RH2	RL2
H	L	RH3	RL3
Working Byte Register		RL4	
Working Word Register		R5	
FA′		R8	
BC′		R9	
DE′		R10	
HL′		R11	
IX		R6	
IY		R7	

Figure 2-5B. Z80 Register Mapping

In figure 2-4, m identifies the source CPU (8080, 8085 or Z80), a identifies the source assembly language (I,5 or Z), and q identifies the current Care/Don't Care status of the translation qualities.

The mapping format shown in figure 2-5A is displayed when the source CPU is either an 8080 or 8085. When the source CPU is a Z80, the format shown in figure 2-5B is displayed to identify the Z80 alternate register set mapping and index registers.

2-13. ACTION: Quit

Entering a Q response causes TRANZ to halt execution, without generating any output. Output generated prior to the issuance of the Q command can be saved by issuing a Delete until End of file command.

2-14. MODE-SETTING COMMANDS

Mode-setting commands enable the user to specify the exactness of the translation through setting Care/Don't Care qualities, and allows specification of the terminal display type as well as the selection of output types to be included in the object file. In Interactive operating mode, the ACTION command prompt is displayed following the utility call sequence. An M response causes TRANZ to enter mode-setting class control, display the prompt

```
  MODE (L,C,D,T,O,I,S,A): __
```

and wait for a response. By entering a delimiter (NEW LINE, blank, or non-alphanumeric character), the user can command a display of mode-setting class options, shown in figure 2-2.

TRANZ continues to prompt for MODE-setting commands until the user specifies a switch to ACTION class commands (MODE command A).

NOTE

All MODE-SETTING commands must be terminated with a delimiter (NEW LINE, blank, or non-alphanumeric character) before they will be accepted by TRANZ for processing.

2-15. MODE: List

Entering an L response causes TRANZ to complete the command line as

```
  MODE (L,C,D,T,O,I,S,A):  LIST
```

and to display the current translation-quality settings and register mapping, as shown in figures 2-4 and 2-5. In figure 2-4, m identifies the source CPU (8080, 8085, or Z80), a identifies the source assembly language (I,5 or Z), and q identifies the current Care/Don't Care status of the translation qualities.

The mapping format shown in figure 2-5A is displayed when the source CPU is either an 8080 or 8085. When the source CPU is a Z80, the format shown in figure 2-5B is displayed to identify the Z80 alternate register set mapping and index registers.

(The LIST command is executed when it is received; a delimiter may not be used to terminate the command.)

2-16. MODE: Care

A Care response specifies that selected translation qualities are to be set to care condition.

```
MODE (L,C,D,T,O,I,S,A): CARE QUALITIES (C,Z,S,P,A,F,D,B,K): __
```

The user can set any or all translation qualities (see figure 2-4) to Care condition to specify that the translated code exactly simulate the function of the source code. Note that the C (Carry), Z (Zero) and S (Sign) flags default to Care conditions; all others default to Don't Care condition. (Refer to chapter 1 for a detailed discussion of the translation quality Care/Don't Care settings and the effects of each on the object code generated.)

2-17. MODE: Don't Care

A Don't Care response specifies that selected translation qualities are to be set (or reset) to don't care condition.

```
MODE (L,C,D,T,O,I,S,A) DONT CARE QUALITIES (C,Z,S,P,A,F,D,B,K): __
```

The user can set default Care qualities C,Z,S to Don't Care or can at any time reset any Care qualities (see figure 2-4) that are no longer required, thus generating smaller, faster-execution object code. (Refer to chapter 1 for a detailed discussion of the Care/Don't Care settings and the effects of each on the object code generated.)

2-18. MODE: Terminal Output Select

A Terminal Output Select response allows the user to specify the types of output to be displayed at the terminal.

```
MODE (L,C,D,T,O,I,S,A): TERMINAL OUTPUT SELECTION (S,O,Z,W,N,C): __
```

Entering a command delimiter causes the selected output types to be displayed at the terminal as they occur.

Source specifies that all source program operations and pseudo-operations, and all source code comments are to be displayed.

Object specifies that all object lines (translated source code) are to be displayed.

Z(error) specifies that all translation errors are to be displayed.

Notes specifies that all translation notes are to be displayed.

Warnings specifies that all translation warnings are to be displayed.

Commands specifies that all translator commands are to be displayed.

The default displays are Notes, Warnings and Commands in Interactive mode, and Warnings in Batch mode (Error messages are always displayed.) To restore the default display-option settings, enter the Terminal display response to the MODE command and enter a command delimiter with no display options. This null-list entry restores the default settings.

2-19. MODE: Object Output Select

The Object Output Select allows the user to specify the types of output to be contained in the object file.

```
MODE (L,C,D,T,O,I,S,A): OBJECT OUTPUT SELECTION (S,O,Z,W,N,C): __
```

Entering a command delimiter causes the selected types to be output to the object file as they occur.

Source specifies that all source program operations and pseudo-operations, and all source code comments are to be output to the object file.

Object specifies that all object lines (translated source code) are to be output to the object file.

Z(error) specifies that all translation errors are to be output to the object file.

Notes specifies that all translation notes are to be output to the object file.

Warnings specifies that all translation warnings are to be output to the object file.

Commands specifies that all translator commands are to be output to the object file.

The default is to output object only. To restore the object output default options, enter the Object select response to the MODE command and enter a command delimiter with no other options. This null-list entry restores the default settings.

2-20. MODE: Input Column Select

The I mode-setting command allows the user to define the source-line format to be translated, by specifying character-column positions. In this way, extraneous characters such as source line numbers, program counter values and binary object listings are ignored in the translation process.

```
  MODE (L,C,D,T,O,I,S,A): INPUT COLUMNS (FROM,TO): __
```

The selected columns are entered as two decimal values (v1<v2) separated with a comma and terminated with any delimiter. TRANZ acknowledges the selection with the display

```
  TRANSLATE FROM nn TO nn
```

To reset the source-line translation limits to translate columns 1 to 80 (the default setting), enter the I response to the MODE command, followed by any delimiter.

2-21. MODE: Store Command String

Store Command String allows the user to specify a command string of up to 40 bytes in length.

```
  MODE (L,C,D,T,O,I,S,A): STORE COMMAND (STRING): __
```

TRANZ halts and waits for the command string to be entered. The stored command can comprise any number of commands, up to the 40-byte limit, and commands can be nested within the boundary parentheses. As an example, a stored command string of the form

```
  (T'OPFLD',R1; MACNAME RL; //X)
```

would direct TRANZ to translate until the string OPFLD is encountered, replace that one line with the line MACNAME RL, and re-execute the stored command. This would effectively translate all lines with the specified opcode field to a macro call.

Note that each command is terminated with a delimiter (in the above example both a comma and a semicolon were used). The entire stored command, including the terminating double slash, must be enclosed in parentheses. Command sets nested within a stored command also must include the terminating double slash and surrounding parentheses. Stored commands are executed using the Action command X (refer to paragraph 2-10 for a description of the eXecute command).

2-22. MODE: Action

An A response causes TRANZ to switch to Action class and prompt for a command.

```
MODE (L,C,D,T,O,I,S,A): ACTION COMMAND

ACTION (T,C,R,D,X,M,L,H): __
```

(This command is executed when it is received; a delimiter may not be used to terminate the ACTION command.) TRANZ remains under Action class control until another M (Mode) command is issued.

2-23. PROGRAM TERMINATION

TRANZ makes two passes through the source code. The actual translation is performed during pass one; address relative corrections are made during pass two. (TRANZ automatically enters pass two upon reaching the end of the source file.) At the end of pass two TRANZ terminates, all files are closed, the temporary file TRANZ.$$$ is deleted, and control returns to the operating system.

The Quit command also can be used to abort TRANZ at any point in the translation cycle. After normal (end-of-source-file) termination, the message shown in figure 2-6 is displayed, where nn are decimal values representing the number of bytes of machine code in the source file and translated object file, the number of lines of source code and the number of notes, warnings and errors generated during the translation process.

```
TRANZ COMPLETE:         SOURCE nn BYTES nn LINES
                        OBJECT nn BYTES

          ERRORS:       nn
        WARNINGS:       nn
           NOTES:       nn
```

Figure 2-6. TRANZ Termination Message

TRANZ can terminate abnormally if, in Batch mode, all embedded commands are completed before the end of the source code is reached. In this case the message

```
***BATCH COMMAND REQUIRED BUT NOT FOUND - FATAL ERROR
```

will be displayed and TRANZ will abort without producing an object file.

If TRANZ aborts abnormally, or if the user exits before translator operations are completed, the temporary file TRANZ.$$$ (used to hold intermediate results of the translation) is not automatically deleted. Then, if the user attempts to restart the utility, TRANZ will not be able to open a temporary file as it still exists on the diskette. In this case, the fatal error message "-3" is issued. The user must delete the existing TRANZ.$$$ and restart the utility.

2-24. DIAGNOSTIC MESSAGES

Three types of diagnostic messages--notes, warnings, and errors--can be generated. By specifying Mode-Setting options, the user can select the types of messages to be included or excluded from the terminal or object file output. Notes are identified with a single asterisk (*), warnings with two asterisks, and errors with three asterisks. Following display of an error message, the offending line is displayed with an exclamation point (!), the source and object program counter locations are displayed, and the current source line is identified.

NOTES:

```
 *   MAXIMUM FIXUP NOT PERFORMED
 *   ADDRESS EXPRESSION CORRECTED
 *   ADDRESS-RELATIVE FIXUP SUSPENDED ACROSS THIS INSTRUCTION
```

WARNINGS:

```
**   WORD BOUNDARY ERROR - BYTE INSERTED
**   EXPRESSION NOT EVALUATED
**   ILLEGAL SYMBOL
**   ABSOLUTE REFERENCE MADE (PCHL, RST, OR NUMERIC ADDRESS)
**   EXACT TRANSLATION NOT AVAILABLE
**   RESERVED WORD USED
```

ERRORS:

COMMAND ERRORS:

```
***  UNABLE TO OPEN FILE
***  ILLEGAL OPERATION
***  ILLEGAL COMMAND
***  ILLEGAL QUALITY
***  ILLEGAL INPUT COLUMNS
***  BATCH COMMAND REQUIRED BUT NOT FOUND - FATAL ERROR
***  TOO MANY LABELS (>250)
***  TOO MANY INSTRUCTIONS (>1800)
***  LABEL SPACE OVERFLOW
```

SOURCE ERRORS:

```
***  SYNTAX UNACCEPTABLE, NO TRANSLATION MADE
***  UNDEFINED OPCODE OR MACRO ENCOUNTERED
***  UNTRANSLATABLE OPCODE ENCOUNTERED
```

2-25. SAMPLE CALLS

The following two examples demonstrate typical call sequences for TRANZ. In the first example, the sequence is entered as a single line; in the second, all commands and translation qualities are specified in response to prompts.

Example 1: The call sequence

```
  >XPAS TRANZ SAGA,,5,B.
```

specifies batch operation translation of source file SAGA of default type .ASM, 8080/8085 assembly language source, with an object file SAGA.ZSC (default object filename since the destination file parameter is omitted). Translation parameters can be entered on a single line of up to 80 characters; in this format, the parameters must be separated with delimiters. Translation parameters are order-dependent; therefore, when a parameter is omitted, a comma must be inserted in the appropriate position to indicate the omission.

For this example, an assumed error occurs. The error message

```
***  UNTRANSLATABLE OPCODE ENCOUNTERED
RIM !       ; erroneous source statement
```

is displayed during translation, followed by the completion message

```
TRANZ COMPLETE:  SOURCE 12345 BYTES    714 LINES
                 OBJECT 17438 BYTES

  NOTES:      83
  WARNINGS:   20
  ERRORS:     1
```

defines the number of bytes in the source and object codes, the number of lines in the source code, and the type and number of messages generated for the object produced.

Example 2: The call sequence

```
  >XPAS TRANZ

8080, 8085 AND Z80 TO Z8000 TRANSLATOR
TRANZ 1.0  15 MAR 79

ENTER SOURCE (DEFAULT IS .ASM): PARABLE

ENTER DESTINATION (DEFAULT FILE IS PARABLE.ZSC): <NEW LINE keystroke>

ENTER SOURCE ASSEMBLY LANGUAGE (I,5,Z): Z

ENTER OPERATION (I,B): I

ACTION (T,C,R,D,X,M,L,H): MODE SETTING

MODE (L,C,D,T,O,I,S,A): DONT CARE QUALITIES (C,Z,S,P,A,F,D,B,K): CSZ

MODE (L,C,D,T,O,I,S,A): ACTION

ACTION (T,C,R,D,X,M,L,H): TRANSLATE UNTIL (E,S,'WD',#): SEVERITY
(N,W,Z):Z
```

specifies an interactive mode translation of source file PARABLE of default type .ASM, 8080/8085 assembly language source code, and default object filename of PARABLE.ZSC. In the example, the object filename is specified by responding to the command line with a delimiter (NEW LINE keystroke). When interactive mode is selected, TRANZ responds with an ACTION command line. The next selection is MODE, to switch to the mode-setting commands. The mode-setting command sets the CSZ qualities to DONT CARE. The comma following the entry is a command delimiter; when this character is encountered, TRANZ responds with another MODE prompt. The next entry commands a switch to ACTION.

The Action command response is Translate until an error of Severity Z is encountered. (Error severity Z is defined as a source statement for which no translated object code is available, or a command error is encountered.)

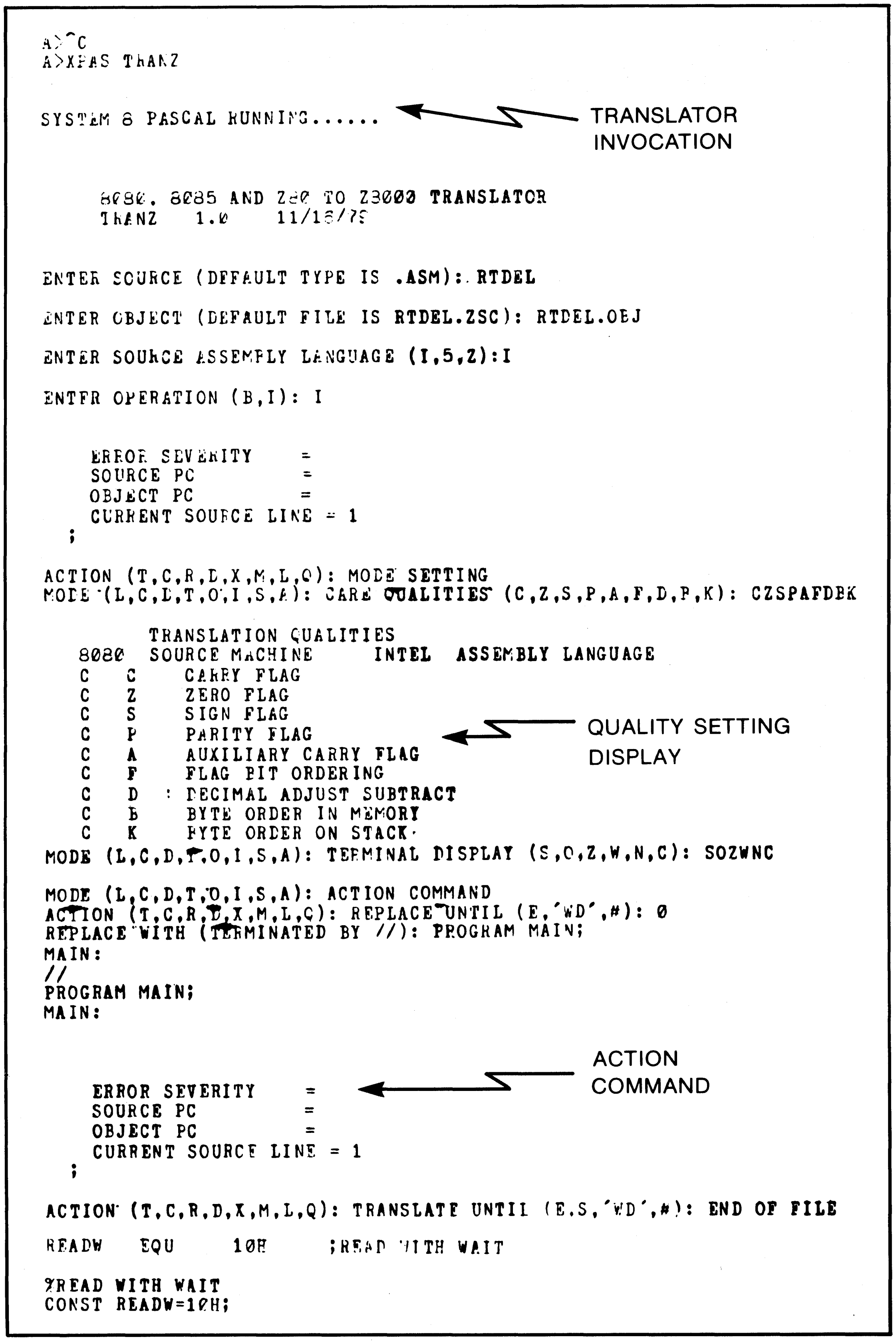
```
*** UNTRANSLATABLE OPCODE ENCOUNTERED
RIM !        ; erroneous source statement


ERROR SEVERITY = Z
SOURCE PC  =  988
OBJECT PC  =  1234
CURRENT SOURCE LINE  =  602
```

```
RIM !         ; erroneous source statement


ACTION (T,C,R,D,X,M,L,H):  DELETE UNTIL (E,'WD',#): END OF FILE


TRANZ COMPLETE:  SOURCE 988  BYTES    602 LINES
                 OBJECT 1234 BYTES

  NOTES:      83
  WARNINGS:   20
  ERRORS:     1
```

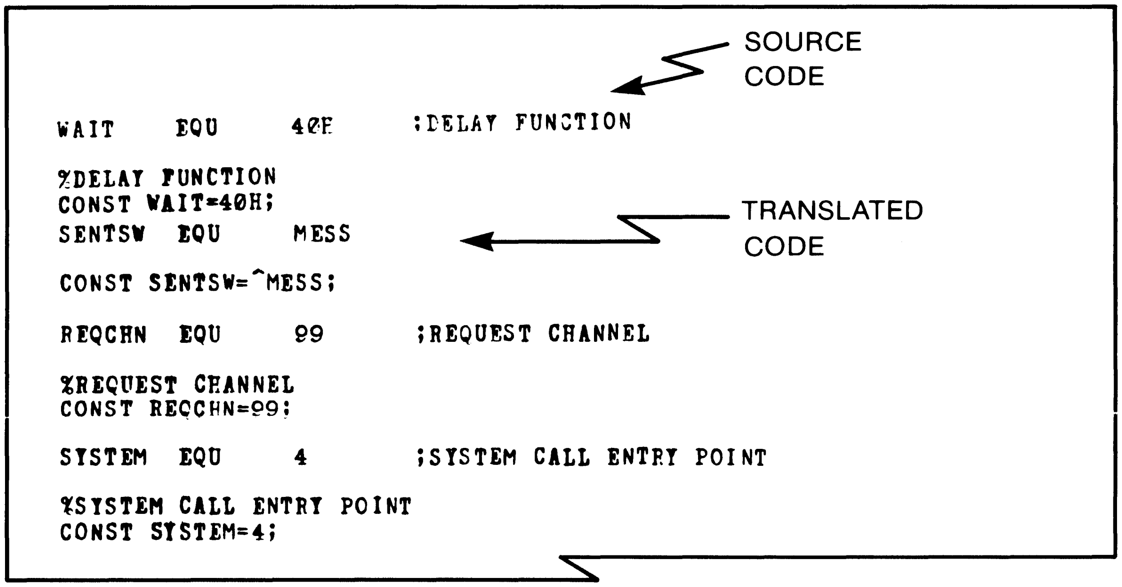
When the Z severity error is encountered, TRANZ displays the erroneous source statement (in the example the 8085 instruction RIM, which is untranslatable since there is no equivalent AmZ8000 instruction) and prompts for another action command. In this case the response is D, or delete (skip) all source instructions from the current line to the end of the file, save all previously translated instructions as the object file, issue the completion message, and exit to the operating system.

2-26. SAMPLE PROGRAM TRANSLATION

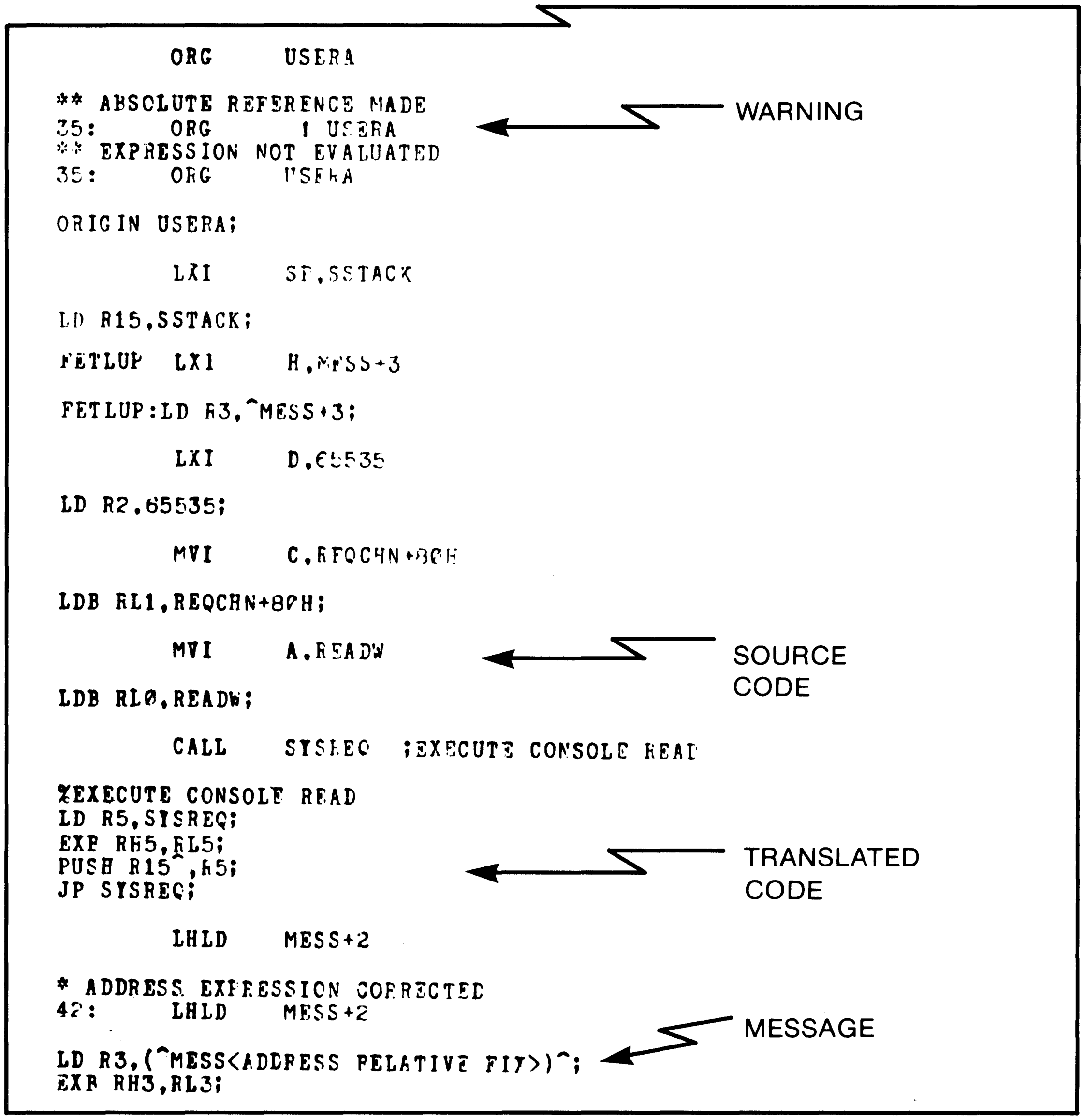
The following pages contain a sample TRANZ translation operation, in which 8080 code is translated into the functionally identical AmZ8000 instructions. The sample is annotated to indicate specific commands, actions and responses during the translate operation.



```
A>^C
A>XPAS TRANZ

SYSTEM 8 PASCAL RUNNING......


     8080, 8085 AND Z80 TO Z8000 TRANSLATOR
     TRANZ    1.0     11/16/79


ENTER SOURCE (DEFAULT TYPE IS .ASM):.RTDEL

ENTER OBJECT (DEFAULT FILE IS RTDEL.ZSC): RTDEL.OBJ

ENTER SOURCE ASSEMBLY LANGUAGE (I,5,Z):I

ENTER OPERATION (B,I): I


    ERROR SEVERITY     =
    SOURCE PC          =
    OBJECT PC          =
    CURRENT SOURCE LINE = 1
  ;

ACTION (T,C,R,D,X,M,L,Q): MODE SETTING
MODE (L,C,D,T,O,I,S,A): CARE QUALITIES (C,Z,S,P,A,F,D,B,K): CZSPAFDBK

         TRANSLATION QUALITIES
   8080  SOURCE MACHINE      INTEL  ASSEMBLY LANGUAGE
   C   C    CARRY FLAG
   C   Z    ZERO FLAG
   C   S    SIGN FLAG
   C   P    PARITY FLAG
   C   A    AUXILIARY CARRY FLAG
   C   F    FLAG BIT ORDERING
   C   D  : DECIMAL ADJUST SUBTRACT
   C   B    BYTE ORDER IN MEMORY
   C   K    BYTE ORDER ON STACK
MODE (L,C,D,T,O,I,S,A): TERMINAL DISPLAY (S,O,Z,W,N,C): SOZWNC

MODE (L,C,D,T,O,I,S,A): ACTION COMMAND
ACTION (T,C,R,D,X,M,L,Q): REPLACE UNTIL (E,'WD',#): 0
REPLACE WITH (TERMINATED BY //): PROGRAM MAIN;
MAIN:
//
PROGRAM MAIN;
MAIN:


    ERROR SEVERITY     =
    SOURCE PC          =
    OBJECT PC          =
    CURRENT SOURCE LINE = 1
  ;

ACTION (T,C,R,D,X,M,L,Q): TRANSLATE UNTIL (E,S,'WD',#): END OF FILE

READW    EQU      10H      ;READ WITH WAIT

?READ WITH WAIT
CONST READW=10H;
```


SOURCE CODE
TRANSLATED CODE

```
WAIT     EQU     40H      ;DELAY FUNCTION

%DELAY FUNCTION
CONST WAIT=40H;
SENTSW   EQU     MESS

CONST SENTSW=^MESS;

REQCHN   EQU     99       ;REQUEST CHANNEL

%REQUEST CHANNEL
CONST REQCHN=99;

SYSTEM   EQU     4        ;SYSTEM CALL ENTRY POINT

%SYSTEM CALL ENTRY POINT
CONST SYSTEM=4;
```


WARNING
SOURCE CODE
TRANSLATED CODE
MESSAGE

```
         ORG     USERA

** ABSOLUTE REFERENCE MADE
35:      ORG      ! USERA
** EXPRESSION NOT EVALUATED
35:      ORG      USERA

ORIGIN USERA;

         LXI     SP,SSTACK

LD R15,SSTACK;

FETLUP   LXI     H,MESS+3

FETLUP:LD R3,^MESS+3;

         LXI     D,65535

LD R2,65535;

         MVI     C,REQCHN+80H

LDB RL1,REQCHN+80H;

         MVI     A,READW

LDB RL0,READW;

         CALL    SYSREQ  ;EXECUTE CONSOLE READ

%EXECUTE CONSOLE READ
LD R5,SYSREQ;
EXB RH5,RL5;
PUSH R15^,R5;
JP SYSREQ;

         LHLD    MESS+2

* ADDRESS EXPRESSION CORRECTED
42:      LHLD    MESS+2

LD R3,(^MESS<ADDRESS RELATIVE FIX>)^;
EXB RH3,RL3;
```

```
        MOV     A,H

LDB RLØ,RH3;

        ORA     L

ORB RLØ,RL3;
RESFLG CY;
LDCTLB RL4,FLAGS;
RESB RL4,3;
RESB RL4,2;
LDCTLB FLAGS,RL4;
```

CALL TO FLAG-BIT ORDERING ROUTINE

```
GETNAA  PUSH    PSW

GETNAA:CALL @@@@F2;
EXB RHØ,RLØ;
PUSH R15^,RØ;
EXB RHØ,RLØ;

% BYPASS ROUTINES

 JR @@@@F12;
@@@@F1: % POP PSW
% CARE ABOUT FLAG BIT ORDER
% SUBROUTINE TO RESTORE FLAGS FROM 8Ø8Ø STYLE PSW TO Z8ØØØ
% CLEAR FLAGS
 CLRB RL4;
% SET DA AND H FLAGS
 BITB RHØ,4;
 JR ZR,(^$+4)^;
 SETB RL4,2;
 LDCTLB FLAGS,RL4;
% RESTORE C
 BITB RHØ,Ø;
% SKIP NEXT INSTRUCTION FOR RESET
 JR ZR,(^$+4)^;
 SETFLG CY;
% RESTORE P
 BITB RHØ,2;
 JR ZR,(^$+4)^;
 SETFLG PY;
% RESTORE S
 BITB RHØ,7;
 JR ZR,(^$+4)^;
 SETFLG SGN;
% RESTORE Z
 BITB RHØ,6;
 RET;
@@@@F2: % PUSH PSW
% CARE ABOUT FLAG BIT ORDER
% SUBROUTINE TO CONVERT Z8ØØØ FLAGS TO 8Ø8Ø IN RHØ
% GET FLAGS BYTE
 LDCTLB RL4,FLAGS;
% CLEAR FLAGS
 CLRB RHØ;
% P FLAG BIT
 JR PO,(^$+4)^;
 SETB RHØ,2;
```

```
% S FLAG BIT
 JR PL,(^$+4)^;
 SETB RH0,7;
% C FLAG BIT
 JR NC,(^$+4)^;
 SETB RH0,0;
% Z FLAG BIT
 JR NZ,(^$+4)^;
 SETB RH0,6;
% SET DEFAULTS
 SETB RH0,1;
% TEST AND SET H FLAG BIT
 BITB RL4,2;
 JR ZR,(^$+4)^;
 SETB RH0,4;
 LDCTLB FLAGS,RL4;
 RET;
@@@@F12! % DONE


          MOV       A,B

LDB RL0,RH1;

          CPI       4

LDB RL4,RL0;
SUBB RL4,4;
ANDB RL4,RL4;

          JM        POPTWO

JP MI,POPTWO;

GETNAH    POP       PSW

GETNAH:POP R0,R15^;                      CALL TO FLAG-BIT
EXB RH0,RL0;                             ORDERING ROUTINE
CALL @@@@F1;

          DCR       B

DECB RH1,1;
TESTB RH1;

          STAX      D

LDB R2^,RL0;

          INX       D

LDCTLB RL4,FLAGS;
INC R2,1;
LDCTLB FLAGS,RL4;
BYTE (14);

SRCE      DS        11         ;SAVES SOURCE FILE NAME

SRCE:%SAVES SOURCE FILE NAME
BYTE (11);

DISKNO    DS        1          ;DEFAULT DISK

DISKNO:%DEFAULT DISK
BYTE (1);
```

```
DISPRM  DS      1         ;UNIT FROM GETNAM

DISPRM:%UNIT FROM GETNAM
BYTE (1);

        DS      26        ;STACK

%STACK
BYTE (26);

        END

END.

  TRANZ COMPLETE: SOURCE   551 BYTES  264 LINES
                  OBJECT   1291 BYTES

      ERRORS:
      WARNINGS: 2
      NOTES:    5
```

COMPLETION MESSAGE

APPENDIX A
ASCII CHARACTER SET

HEX	DEC	CHAR	HEX	DEC	CHAR	HEX	DEC	CHAR	HEX	DEC	CHAR
00	0	NUL	20	32	SP	40	64	@	60	96	
01	1	SOH	21	33	!	41	65	A	61	97	a
02	2	STX	22	34	"	42	66	B	62	98	b
03	3	ETX	23	35	#	43	67	C	63	99	c
04	4	EOT	24	36	$	44	68	D	64	100	d
05	5	ENQ	25	37	%	45	69	E	65	101	e
06	6	ACK	26	38	&	46	70	F	66	102	f
07	7	BEL	27	39	'	47	71	G	67	103	g
08	8	BS	28	40	(	48	72	H	68	104	h
09	9	HT	29	41	)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	|
1D	29	GS	3D	61	=	5D	93	]	7D	125	}
1E	30	RS	3E	62	>	5E	94	^	7E	126	~
1F	31	US	3F	63	?	5F	95	_	7F	127	DEL

APPENDIX B
FLAG-BIT ORDERING ROUTINES

This appendix contains the flag-bit ordering routines, @@@@F1 and @@@@F2, that are added to the translated output code when the F translation quality is set to care condition. With this care setting, the bits in the Flag Control Word (FCW) are exactly simulated in the output code. Note that the Care/Don´t Care status of the F quality must be identical for the PUSH/POP and CALL/RET instructions.

TABLE B-1. FLAG FIX-UP ROUTINES

```
% BYPASS ROUTINES
  JR @@@@F12;
@@@@F1:  % POP PSW
% CARE ABOUT FLAG BIT ORDER
% SUBROUTINE TO RESTORE FLAGS FROM SOURCE PSW TO Z8000
% CLEAR FLAGS
 CLRB RL4;
% SET DA AND H FLAGS
 BITB RHØ,4;
 JR ZR, (↑$+4)↑;
 SETB RL4,2;
 LDCTLB FLAGS,RL4;
% RESTORE C
 BITB RHØ,Ø;
% SKIP NEXT INSTRUCTION FOR RESET
 JR ZR, (↑$+4)↑;
 SETFLG CY;
% RESTORE P
 BITB RHØ,2;
 JR ZR,(↑$+4)↑;
 SETFLG PY;
% RESTORE S
 BITB RHØ,7;
 JR ZR,(↑$+4)↑;
 SETFLG SGN;
% RESTORE Z
 BITB RHØ,6;
 RET;
```

```
@@@@F2:  % PUSH PSW
% CARE ABOUT FLAG BIT ORDER
% SUBROUTINE TO CONVERT Z8000 FLAGS TO SOURCE FORMAT IN RHØ
% GET FLAGS BYTE
 LDCTLB RL4,FLAGS;
% CLEAR FLAGS
 CLRB RHØ;
% P FLAG BIT
 JR PO,(↑$+4)↑;
 SETB RHØ,2;
% S FLAG BIT
 JR PL,(↑$+4)↑;
 SETB RHØ,7;
% C FLAG BIT
 JR NC,(↑$+4)↑;
 SETB RHØ,Ø;
% Z FLAG BIT
 JR NZ,(↑$+4)↑;
 SETB RHØ,6;
% SET DEFAULTS
 SETB RHØ,1;
% TEST AND SET H FLAG BIT
 BITB RL4,2;
 JR ZR,(↑$+4)↑;
 SETB RHØ,4;
 LDCTLB FLAGS, RL4;
RET;
@@@@F12:  % DONE
```

APPENDIX C
PSEUDO-OPERATION EQUIVALENCIES

Table C-1 defines the directive equivalencies of the Am8080, Intel 8080/8085, Z80 and AmZ8000 assemblers. Operators that can be used in forming expressions are defined with the precedence of evaluation listed parenthetically following each operator. The 8080, Intel 8080/8085 and Z80 source statements are all MACRO8 (MAC) assembler compatible; the AmZ8000 statements are MACRO8000 (MACZ) assembler compatible.

TABLE C-1. PSEUDO-OPERATION EQUIVALENCIES

PSEUDO-OPs	AMC 8080	INTEL 8080/85	Z80	AmZ8000
ORIGIN	ORG (or .PHASE<addr> ... .DEPHASE)	ORG	ORG	ORIGIN
EXTERNAL	Label followed by "##" or EXTRN or EXT	EXTRN	EXTERNAL	EXTERNAL
GLOBAL	Label followed by "::" or PUBLIC or ENTRY	PUBLIC	GLOBAL	GLOBAL
SET variable	SET	SET	DEFL	VAR X: OBJECT; X::=...
EQU constant	EQU	EQU	EQU	CONST X=...;
END	END	END	END	END.
allocate storage	DS	DS	DEFS	BYTE (n)
count: string			DEFT '...'	STRING: '...'
string: EOS	DC			
string of characters	DB	DB	DEFM	BYTE:
word • reverse store of bytes	DW • values and characters reversed	DW • all reverse	DEFW • reverse store values only	WORD: WORD:SWAP(X)
byte	DB	DB	DEFB	BYTE:
absolute PC	ASEG	ASEG (default)	(default)	ABSOLUTE
Data PC	DSEG	DSEG ƀ PAGE INPAGE		SEGMENT [@CUM],'DSEG';
Code PC	CSEG(default)	CSEG ƀ PAGE INPAGE		SEGMENT [@CUM],'CSEG'
named or blank "common" segments	COMMON			SEGMENT[@COM]

Stack Length		STKLN		SEGMENT
				[@COM],'STACK';
				BYTE (n);
				SEGMENT @PRIOR;

Comments	;...<cr> or.COMMENT <delim> ...<delim>	;...<cr>	;...<cr>	%...<cr> Or (*... ...*)
display "MOUNT TAPE" wait for "ƀ" entry		EOT		

TABLE C-1. PSEUDO-OPERATION EQUIVALENCIES (Cont.)

PSEUDO-OPs	AMC 8080	INTEL 8080/85	Z80	AmZ8000
set default radix	.RADIX r r is 2...16			(see "specified radix")
OPERANDS				
BO Stack		STACK		EXTERNAL /STACK/
EO Program +1		MEMORY		EXTERNAL /HIGH ADDRESS/
binary	n...B	n...B	n...B	n...B
octal	n...O or Q	n...O or Q	n...O or Q	n...O or Q
decimal	n...D or n...	n...D or n...	n...D or n...	n...D or n...
hex	n...H or X′xxx′	n...H	n...H	n...H or #n...
specified radix				r#n... r is 2..16
representation	8 or 16 bits	8 or 16 bits	16 bits	variable
Literals:	<escape> or !escape "..." or ′...′	!escape ′...′ ′<—′′′′	′...′ ′<—′′′′ 0="	 ′...′ <—′′′′ 0="
Line # (source line header)	Line # (high bit on)			
Identifiers				
Labels case	UC = lc	UC = lc	UC ≠lc	UC = 1C
Labels length	6 characters	6 characters	6 characters	unlimited
lead char.	$|.|?|@|a ...	?..., @..., a...		@...., a...
ops labels	trailing colon	trailing colon	trailing colon or character in column 1	trailing colon
pseudo-op labels	trailing colon	trailing blank	trailing colon or character in column 1	trailing colon
legal characters (Identifiers are non-reserved words)	A...Z 0...9 $,.,?,@	A...Z 0...9 ?	A...Z 0...9 ? _ (underscore)	A...z 0...9 @ _ (underscore)
ignore				ignore embedded _ in numbers
addresses PC	expression $	expression $	expression $	expression↑ $
statement separator	<cr>	<cr>	<cr>	;
listing options	.LIST .XLIST .LALL, .SALL, .XALL, .CREF, .XCREF $EJECT PAGE [size]		*List ON *List OFF *Maclist ON *Maclist OFF *Eject	LIST NOLIST EJECT PAGE n

TABLE C-1. PSEUDO-OPERATION EQUIVALENCIES (Cont.)

PSEUDO-OPS	AMC 8080	INTEL 8080/85	Z80	AmZ8000
Name	NAME ('...')	NAME	*Heading	MODULE '...'
Title	[$]TITLE '... SUBTTL '...'			TITLE: '...'
include			*Include	INCLUDE
Terminal Output	.PRINTX<delim> ...<delim>			
MACROS	MACRO	MACRO	MACRO	MACRO
Parameters	standard symbol	standard symbol	leading #	standard symbol
	ENDM	ENDM	ENDM	END;
	EXITM	EXITM		EXIT;
	LOCAL	LOCAL		VAR x:object;
	REPT exp	REPT		FOR exp DO BEGIN
	IRP var,<v_1...>	IRP		FOR var IN (v_1...) DO BEGIN
	IRPC var,<...> or IRPC var...	IRPC		FOR var IN '...' DO BEGIN
Macro Library:	.REQUEST			
User Symbol Generation:				
evaluate		%		
concatenate	& (before user symbol only)	&(before or after user symbol)		
counter			#$YM Incrementing counter	
Conditional Assembly				
if true	IF or IFT	IF	COND	IF...THEN BEGIN If NOT...
if false	IFE or IFF			
else	ELSE	ELSE		END ELSE BEGIN
end	ENDIF	ENDIF		END
Conditionals				
Pass 1	IF1			
Pass 2	IF2			
defined	IFDEF [not (IFNDEF)]			
blank expression	IFB [not (IFNB)]			
F=0 T≠0 (paren allowed)				

TABLE C-1. PSEUDO-OPERATION EQUIVALENCIES (Cont.)

PSEUDO-OPs	AMC 8080	INTEL 8080/85	Z80	AmZ8000
OPERATORS				
null	NUL$^{(1)*}$	NUL$^{(1)*}$		NULL $^{(1)*}$
High byte	HIGH$^{(2)}$	HIGH$^{(2)}$		HIGH$^{(2)}$
Low Byte	LOW$^{(2)}$	LOW$^{(2)}$		LOW$^{(2)}$
*	*$^{(3)}$	*$^{(3)}$	*$^{(3)}$	*$^{(3)}$
/	/$^{(3)}$	/$^{(3)}$	/$^{(3)}$	/$^{(3)}$
MOD	MOD$^{(3)}$	MOD$^{(3)}$	.MOD.$^{(3)}$	MOD$^{(3)}$
shift right	SHR$^{(3)}$	SHR$^{(3)}$	.SHR.$^{(3)}$	SHR$^{(3)}$
shift left	SHL$^{(3)}$	SHL$^{(3)}$	.SHL.$^{(3)}$	SHL$^{(3)}$
- unary	- unary$^{(4)}$	- unary$^{(4)}$	- unary$^{(4)}$	- unary$^{(4)}$
+	+$^{(5)}$	+$^{(4)}$	+$^{(4)}$	+$^{(5)}$
-	-$^{(5)}$	-$^{(4)}$	-$^{(4)}$	-$^{(5)}$
+ unary	+ unary$^{(5)}$	+ unary$^{(4)}$	+ unary(1)	+ unary $^{(4)}$
NE	NE$^{(6)}$	NE$^{(5)}$		NE$^{(6)}$
LE	LE$^{(6)}$	LE$^{(5)}$		LE$^{(6)}$
GE	GE$^{(6)}$	GE$^{(5)}$		GE$^{(6)}$
EQ	EQ$^{(6)}$	EQ$^{(5)}$	.EQ.≡'='$^{(7)}$	EQ$^{(6)}$
GT	GT$^{(6)}$	GT$^{(5)}$	.GT.≡'>'$^{(7)}$	GT$^{(6)}$
LT	LT$^{(6)}$	LT$^{(5)}$	.LT.≡'<'$^{(7)}$	LT$^{(6)}$
NOT	NOT$^{(7)}$	NOT$^{(7)}$	.NOT.≡'/'$^{(1)}$	NOT$^{(7)}$
AND	AND$^{(8)}$	AND$^{(7)}$	.AND.≡'&'(5)	AND$^{(8)}$
OR	OR$^{(9)}$	OR$^{(8)}$	.OR.≡'↑'$^{(6)}$ .XOR.$^{(6)}$	OR$^{(9)}$
XOR	XOR$^{(9)}$	XOR$^{(8)}$		XOR$^{(9)}$
residue - after ±overflow			.RES.$^{(1)}$	
**			**(2)	
unsigned GT			.UGT.$^{(7)}$	LGT$^{(6)}$
unsigned GE				LGE$^{(6)}$
unsigned LT			.ULT.$^{(7)}$	LLT$^{(6)}$
unsigned LE				LLE$^{(6)}$

NOTE
*Parenthetical numbers indicate operator precedence in each language.

APPENDIX D
8080/8085 CODE TRANSLATION SUMMARY

Table D summarizes the object code generated by TRANZ for each 8080/ 8085 source code line. TRANZ allows the user to specify, through a series of prompts, Care or Don't Care conditions for the selected translation qualities. As shown the object code generated differs, depending on the setting of qualities. With some instructions, up to five lines of object code (14 bytes) can be generated to satisfy the Care-code settings specified by the user.

The table also defines the FCB flag bits affected by the instruction; i.e., bits modified or set to a specific value to reflect the results of the operation. A Ø or 1 entry in the flags-affected column indicates the constant value of the related bit following execution of the instruction; the flag code letter in the entry indicates that the flag is used to define the CPU condition following execution, and a blank indicates that the flag setting is unchanged.

The FCB flag bits and translation qualities are defined as:

C	= Carry
Z	= Zero
S	= Sign
P	= Parity (same as Z80/AmZ8000 P/V, Parity/Overflow)
A	= Auxiliary Carry (AC, same as Z80/AmZ8000 H, Half Carry)
B	= Byte Swap (word/byte reference order reversal)
K	= Stack Byte Swap (word/byte stack reference order reversal)
F	= Flag Bit ordering
D	= Decimal Adjust (same as Z80/Z8000 N flag)

Within the instructions, the following abbreviations and symbols are used:

br	= byte register
bv	= byte value
mbr	= mapped byte register
mwr	= mapped word register

wr	=	word register
wv	=	word value
(X)	=	contents of X
$(\overline{X})$	=	complement of contents of X
((X))	=	contents of contents of X. Contents of designated location in inner brackets are interpreted as address to a second location that contains the 8-bit operand.
↑X	=	address of X
X↑	=	item referenced by the address of X
<—	=	receives
<—>	=	is exchanged with
/\	=	logical AND
V	=	logical OR
⊻	=	logical EXCLUSIVE OR

Note: Unless otherwise specified in the table, translated instructions are modified for AmZ8000 compability. In some cases, the modified inexact object code is flagged with a message to that effect. Where only the length of the code is changed, a change message is not issued. Unless flagged, the resultant object code is functionally identical to the source code.

TABLE D-1. 8080/8085 TRANZ CODE TRANSLATION SUMMARY

8080/8085 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH

ACI bv	2	A<—(A)+(bv)+(c)	C Z S P AC	LDB RL4,bv;
				ADCB RLØ,wbr;
					4	P	LDB RL4,bv;
							ADCB RL0,RL4;
							ANDB RL0,RL0;
								6

ADC br	1	A<—(A)+(br)+(C)	C Z S P AC	ADCB RL0,mbr;	2	P	ADCB RL0,mbr;
							ANDB RL0,RL0;
								4

ADC M	1	A<—(A)+((M))+(C)	C Z S P AC	LDB RL4,R3↑;
				ADDB RL0,R3↑;
					4	P	LDB RL4,R3↑;
							ADCB RL0, RL4;
							ANDB RL0,RL0;
								6

ADD br	1	A<—(A)+(br)	C Z S P AC	ADDB RL0,mbr;	2	P	ADDB RL0,mbr;
							ANDB RL0,RL0;
								4

ADD M	1	A<—(A)+((M)	C Z S P AC	ADDB RL0,R3↑;	2	P	ADDB RL0,R3↑; ANDB RL0,RL0;	4

ADI bv	2	A<—(A)+(bv)	C Z S P AC	ADDB RL0,bv;	4	P	ADDB RL0,bv;
							ANDB RL0,RL0;
								6

ANA br
(8080
only)
	1	A<—(A)/\(br)	0 Z S P 0	ANDB RL0,mbr;	2	C	ANDB RL0,mbr;
							RESFLG CY;
								2

						AC	ANDB RL0,mbr;
							LDCTLB RL4,FLAGS;
							RESB RL4,3;
							RESB RL4,2;
							LDCTLB FLAGS,RL4;
								10

ANA br
(8085 only)
	1	A<—(A)/\(br)	0 Z S P 1	ANDB RL0,mbr;	2	C	ANDB RL0,mbr;
							RESFLG CY;
								4

						AC	ANDB RL0,mbr;
							LDCTLB RL4,FLAGS;
							RESB 3,RL4;
							SETB 2,RL4;
							LDCTLB FLAGS,RL4;
								10

ANA M (8080 only)	1	A<—(A)/\((M))	0 Z S P 0	ANDB RL0,R3↑;	2	C	ANDB RL0,R3↑; RESFLG CY;	4

						AC	ANDB RL0,R3↑;
							LDCTLB RL4,FLAGS;
							RESB 3,RL4;
							RESB 2,RL4;
							LDCTLB FLAGS,RL4;
								10

TABLE D-1. 8080/8085 TRANZ CODE TRANSLATION SUMMARY (Cont.)

8080/8085 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH
ANA M (8085 only)	1	A←(A)∧(M)	0 Z S P 1	ANDB RL0,R3↑;	2	C	ANDB RL0,R3↑; RESFLG CY;	4

						AC	ANDB RL0,R3↑;
							LDCTLB RL4,FLAGS;
							RESB 3,RL4;
							SETB 2,RL4;
							LDCTLB FLAGS,RL4;
								10

ANI bv
(8080 only)
	2	A←(A)∧(bv)	0 Z S P 0	ANDB RL0,bv;	4	C	ANDB RL0,bv;
							RESFLG CY;
								6

						AC	ANDB RL0,bv;
							LDCTLB RL4,FLAGS;
							RESB RL4,3;
							RESB RL4,2;
							LDCTLB FLAGS,RL4;
								12

ANI bv
(8085 only)
	2	A←(A)∧(bv)	0 Z S P 1	ANDB RL0,bv;	4	C	ANDB RL0,bv;
							RESFLG CY;
								6

						AC	ANDB RL0,bv;
							LDCTLB RL4,FLAGS;
							RESB RL4,3;
							SETB RL4,2;
							LDCTLB FLAGS,RL4;
								12

CALL addr	3	SP←SP-2
		(SP)←PC
		PC←addr
				CALL addr;	4	K	LD R5,wv;
							EXB RH5,RL5;
							PUSH R15↑,R5;
							JP addr;
								12

CC addr	3	IF condition
		THEN CALL
				JR NC,(↑$+6)↑;
				CALL addr;
					6	K	JR NC,(↑$+14)↑; LD
							R5,addr;
							EXB RH5,RL5;
							PUSH R15↑,R5;
							JP addr;
								14

CNC addr	3			JR CY,(↑$+6)↑;
				CALL addr;
					6	K	JR CY,(↑$+14)↑; LD
							R5,addr;
							EXB RH5,RL5;
							PUSH R15↑,R5;
							JP addr;
								14

TABLE D-1. 8080/8085 TRANZ CODE TRANSLATION SUMMARY (Cont.)

8080/8085 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH
CP addr	3			JR MI,(↑$+6)↑; CALL addr;	6	K	JR MI,(↑$+14)↑; LD R5,addr; EXB RH5,RL5; PUSH R15↑,R5; JP addr;	14
CM addr	3			JR PL,(↑$+6)↑; CALL addr;	6	K	JR PL,(↑$+14)↑; LD R5,addr; EXB RH5,RL5; PUSH R15↑,R5; JP addr;	14
CPE	3	IF condition THEN CALL		JR PO,(↑$+6)↑; CALL addr;	6	K	JR PO,(↑$+14)↑; LD R5,addr; EXB RH5,RL5; PUSH R15↑,R5; JP addr;	14
CPO addr	3			JR PE,(↑$+6)↑; CALL addr;	6	K	JR PE,(↑$+14)↑; LD R5,addr; EXB RH5,RL5; PUSH R15↑,R5; JP addr;	14
CZ addr	3			JR NZ,(↑$+6)↑; CALL addr;	6	K	JR NZ,(↑$+14)↑; LD R5,addr; EXB RH5,RL5; PUSH R15↑,R5; JP addr;	14
CNZ addr	3	IF condition THEN CALL		JR Z,(↑$+6)↑; CALL addr	6	K	JR Z,(↑$+14)↑; LD R5,addr; EXB RH5,RL5; PUSH R15↑,R5; JP addr;	14
CMA	1	A<—($\overline{A}$)		COMB RL0;	2	S or Z or P	LDCTLB RL4,FLAGS; COMB RL0; LDCTLB FLAGS,RL4;	6
CMC	1	C<—($\overline{C}$)	C	COMFLG C;	2		N/A	

TABLE D-1. 8080/8085 TRANZ CODE TRANSLATION SUMMARY (Cont.)

8080/8085 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH

CMP br	1	C,Z<—(A)-(br)	C Z S P AC	CPB RL0,mbr;	2	P
						or
						AC
							LDB RL4,RL0;
							SUBB RL4,mbr;
							ANDB RL4,RL4;
								6

CMP M	1	C,Z<—(A)-(M)	C Z S P AC	CPB RL0,R3↑;	2	P
						or
						AC
							LDB RL4,R3↑;
							SUBB RL4,RL0;
							ANDB RL4,RL4;
								6

CPI bv	2	C,Z<—(A)-(bv)	C Z S P AC	CPB RL0,bv;	4	P
						or
						AC
							LDB RL4,bv;
							SUBB RL4, RL0;
							ANDB RL4,RL4;
								6

DAA	1	A<—2BCD digits	C Z S P AC+	DAB RL0;	2	P	DAB RL0;
							ANDB RL0,RL0;
								4

						D	LDCTLB, RL4,FLAGS;
							SETB RL4,3;
							LDCTLB FLAGS,RL4;
							DAB RL0;
								8*

DAD wr	1	M<—(M)+(wr)	C	ADD RH3,mwr;	2	Z
						or
						S
						or
						P
						or
						AC
							RESFLG C;
							LDCTLB wr, FLAGS;
							ADD R3, mwr;
							JR NC(↑$+2)↑;
							SETB 7,RL4;
							LDCTLB FLAGS,br;
								12

DCR br	1	br<—(br)-1	Z S P AC	DECB mbr,1;	2	P	DECB mbr,1;
							TESTB mbr;
								4

						A	LDCTLB RL4,FLAGS;
							SUBB mbr,1;
							RESFLG CY;
							LDCTLB RL5,FLAGS
							BITB RL4,7;
							JR ZR, (^$+4)^;
							SETB RL5,7;
							LDCTLB FLAGS,RL5;
								16

DCR M	1	(M)<—((M))-1	Z S P AC	DECB R3↑,1;	2	P	DECB R3↑,1; TESTB R3↑;	4

*Translation is not exact and is not modifiable. Warning is issued.

TABLE D-1. 8080/8085 TRANZ CODE TRANSLATION SUMMARY (Cont.)

8080/8085 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH
DCR M (continued)						A	LDCTLB RL4,FLAGS; SUBB R3↑,1; RESFLG CY; LDCTLB RL5,FLAGS BITB RL4,7; JR ZR, (^$+4)^; SETB RL5,7; LDCTLB FLAGS,RL5;	16
DCX wr	1	wr<—(wr)-1		DEC mwr,1;	2	Z or S or P	LDCTLB RL4,FLAGS; DEC mwr,1; LDCTLB RL4,FLAGS;	6
DI	1	Interrupt enable<—0		DI VI,NVI;	2		N/A	
EI	1	Interrupt enable<—1		EI VI,NVI;	2		N/A	
HLT	1	halt		HALT;	2		N/A	
IN bv	2	A<— (port (bv))		INB RL0,bv;	4		*	
INR br	1	br< —(br)+1	Z S P AC	INCB mbr,1;	2	P	INCB mbr,1; TESTB mbr;	4
						A	LDCTLB RL4,FLAGS; ADDB mbr,1; RESFLG CY; LDCTLB RL5,FLAGS; BITB RL4,7; JR ZR, (^$+4)^; SETB RL5,7; LDCTLB FLAGS,RL5	16
INR M	1	(M)<—(M)+1	Z S P AC	INCB R3↑,1;	2	P	INCB R3↑,1; TESTB R3↑;	4

*Translation is not exact for hardware. Program need not be modified. Warning is issued.

TABLE D-1. 8080/8085 TRANZ CODE TRANSLATION SUMMARY (Cont.)

8080/8085 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH

INR M
(continued)
						A	LDCTLB RL4,FLAGS;
							ADDB R3↑,1;
							RESFLG CY;
							LDCTLB RL5,FLAGS;
							BITB RL4,7;
							JR ZR, (^$+4)^;
							SETB RL5,7;
							LDCTLB FLAGS,RL5;
								16

INX wr	1	wr<—(wr)+1		INC mwr,1;	2	Z
						or
						S
						or
						P
							LDCTLB RL4,FLGR;
							INC mwr,1;
							LDCTLB FLAGS,RL4;
								6

JC addr	3			JP CY,addr;	4	N/A		
JNC addr	3			JP NC,addr;	4	N/A		
JP addr	3			JP PL,addr;	4	N/A		

JM addr	3		IF condition
			THEN JMP
				JP MI,addr;	4	N/A		

JPE addr	3			JP PE,addr;	4	N/A		
JPO addr	3			JP PO,addr;	4	N/A		
JZ addr	3	If condition THEN JMP		JP Z,addr;	4		N/A	
JNZ addr	3			JP NZ,addr;	4		N/A	
JMP addr	3	PC<—(addr)		JP addr;	4		N/A	
LDA addr	3	A<—((addr))		LDB RL0,addr↑;	4		N/A	
LDAX wr	1	A<—((wr))		LDB RL0,mwr↑;	2		N/A	
LHLD addr	3	M<—((addr))		LD mhl,addr↑;	4	B	LD R3,addr↑; EXB RH3,RL3;	6
LXI wr,wv	3	wr<—(wv)		LD mwr,wv;	4		N/A	

TABLE D-1. 8080/8085 TRANZ CODE TRANSLATION SUMMARY (Cont.)

8080/8085 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH
POP wr	1	wr<—(SP) SP<—SP+2		POP mwr,R15↑	2	K	POP mwr,R15↑ EXB mwrl,mwrh	4
POP PSW	1		C Z S P AC	POP R0,R15↑; LDCTLB FLAGS,RH0;	4	F	POP R0,R15↑; CALL @@@F1;	6

						K	POP R0,R15↑;
							EXB mwrl,mwrh;
							LDCTLB FLAGS,RH0;
								6

PUSH wr	1	SP<—SP-2
		(SP)<—wr
				PUSH R15↑,mwr;	2	K	EXB mwrl,mwrh;
							PUSH R15↑,mwr;
							EXB mwrl,mwrh;
								6

PUSH PSW	1			LDCTLB RH0,FLAGS;
				PUSH R15↑,R0;
					4	F	CALL @@@F2;
							PUSH R15↑,R0;
								6

						K	LDCTLB RH0,FLAGS;
							EXB RH0,RL0;
							PUSH R15↑,R0;
							EXB RH0,RL0;
								8

RAL	1	$A_{(n+1)}$<—(A_n)
		C<—(A_7)
		A_0<—(C)
			C	RLCB RL0,1;	2	Z
						or
						S
						or
						P
							LDCTLB RL4, FLAGS;
							RLCB RL0,1;
							JR NC,(↑$+6)↑;
							SETB RL4,7;
							JR (↑$+4)↑;
							RESB RL4,7;
							LDCTLB FLAGS,RL4;
								14

RAR	1	An<—(An+1)
		C<—(A_0)
		A_7<—(C)
			C	RRCB RL0,1;	2	Z
						or
						S
						or
						P
							LDCTLB RL4,FLAGS;
							RRCB RL0,1;
							JR NC,(↑$+6)↑;
							SETB RL4,7;
							JR (↑$+4)↑;
							RESB RL4,7;
							LDCTLB FLAGS,RL4;
								14

RC	1			RET C;	2	K	JR NC,(↑$+8)↑; POP R5,R15↑; EXB RH5,RL5; JP R5↑;	8

TABLE D-1. 8080/8085 TRANZ CODE TRANSLATION SUMMARY (Cont.)

8080/8085 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH
MOV br1,br2	1	br1<—(br2)		LDB mbr1,mbr2;	2		N/A	
MOV M,br	1	(M)<—(br)		LDB R3↑,mbr;	2		N/A	
MOV br,M	1	br<—((M))		LDB mbr,R3↑;	2		N/A	
MVI br,bv	2	br<—(bv)		LDB mbr,bv;	4		N/A	
MVI M,bv	2	(M)<—((bv))		LDB R3↑,bv;	4		N/A	
NOP	1	no-operation		NOP;	2		N/A	

ORA br	1	A<—(A) V (br)	0 Z S P 0	ORB RL0,mbr;	2	C	ORB RL0,mbr;
							RESFLG CY;
								4

						AC	ORB RL0,mbr;
							LDCTLB RL4,FLAGS;
							RESB 3,RL4;
							RESB 2,RL4;
							LDCTLB FLAGS,RL4;
								10

ORA M	1	A<—A V (M) fix AC flag after logical op.	0 Z S P 0	ORB RL0,R3↑;	2	C	ORB RL0,R3↑; RESFLAG CY;	4

						AC	ORB RL0,R3↑;
							LDCTLB RL4,FLAGS;
							RESB RL4,3;
							RESB RL4,2;
							LDCTLB FLAGS,RL4;
								10

ORI bv	2	A<—(A) V ((bv))	0 Z S P 0	ORB RL0,bv;	4	C	ORB RL0,bv;
							RESFLG CY;
								6

						AC	ORB RL0,bv;
							LDCTLB RL4,FLAGS;
							RESB RL4,3;
							RESB RL4,2;
							LDCTLB FLAGS,RL4;
								12

OUT bv	2	port (bv)<—(A)		OUTB bv,RL0;	4			*
PCHL	1	PC<—(HL)		JP R3↑;	2		N/A	

*Translation is not exact for hardware. Program need not be modified. Warning is issued.

TABLE D-1. 8080/8085 TRANZ CODE TRANSLATION SUMMARY (Cont.)

8080/8085 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH
RNC	1			RET NC;	2	K	JR CY,(↑$+8)↑; POP R5,R15↑; EXB RH5,RL5; JP R5↑;	8

RP	1	IF condition
		THEN RETURN
				RET PL;	2	K	JR MI,(↑$+8)↑;
							POP R5,R15↑;
							EXB RH5,RL5;
							JP R5↑;
								8

RM	1			RET MI;	2	K	JR PL,(↑$+8)↑; POP R5,R15↑; EXB RH5,RL5; JP R5↑;	8
RPE	1			RET PE;	2	K	JR PO,(↑$+8)↑; POP R5, R15↑; EXB RH5,RL5; JP R5↑;	8
RPO	1			RET PO;	2	K	JR PE,(↑$+8)↑; POP R5,R15↑; EXB RH5,RL5; JP R5↑;	8

RZ	1	IF condition
		THEN RETURN
				RET Z;	2	K	JR NZ,(↑$+8)↑;
							POP R5,R15↑;
							EXB R5,RL5;
							JP R5↑;
								8

RNZ	1			RET NZ;	2	K	JR Z,(↑$+8)↑; POP R5,R15↑; EXB RH5,RL5; JP R5↑;	8
RET	1	PC<—((SP)) PCH<—((SP)+1) SP<—(SP)+2		RET;	2	K	POP R5,R15↑; EXB RH5,RL5; JP R5↑;	6

TABLE D-1. 8080/8085 TRANZ CODE TRANSLATION SUMMARY (Cont.)

8080/8085 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH

RLC	1	$A_{(n+1)}$<—(A_n)
		A_0<—(A_7)
		C<—(A_7)
			C	RLB RL0,1;	2	Z
						or
						S
						or
						P
							LDCTLB RL4,FLAGS;
							RLB RL0,1;
							JR NC,(↑$+6)↑;
							SETB RL4,7;
							JR (↑$+4)↑;
							RESB RL4,7;
							LDCTLB FLAGS,RL4;
								14

RRC	1	A_N<—(A(n+1))
		A_n<—(A_0)
		C<—(A_0)
			C	RRB RL0,1;	2	Z
						or
						S
						or
						P
							LDCTLB RL4,FLAGS;
							RRB RL0,1;
							JR NC,(↑$+6)↑;
							SETB RL4,7;
							JR (↑$+4)↑;
							RESB RL4,7;
							LDCTLB FLAGS,RL4;
								14

RST n	1	$0\underline{<}n\underline{<}7$ SP<—SP-2 (SP)<—PC PC<—n+8		CALL(n*8)↑;	4		N/A	

SBB br	1	A<—(A)-(br)-C	C Z S P AC	SBCB RL0,mbr;	2	P	SBCB RL0,mbr;
							ANDB RL0,RL0;
								4

SBB M	1	A<—(A)-(M)-C	C Z S P AC	LDB RL4,R3↑
				SBCB RL0,RL4;
					4	P	LDB RL4,R3↑;
							SBCB RL0,RL4;
							ANDB RL0,RL0;
								6

SBI bv	2	A<—(A)-(bv)-C	C Z S P AC	LDB RL4,bv;
				SBCB RL0,RL4;
					4	P	LDB RL4,bv;
							SBCB RL0,RL4;
							ANDB RL0,RL0;
								6

SHLD addr	3	(addr)<—HL		LD addr↑,R3;	4	B	EXB RH3,ml;
							LD addr↑,R3;
							EXB RH3,ml;
								8

SPHL	1	SP<—(HL)		LD R15,R3;	2		N/A	
STA addr	3	(addr)<—(A)		LDB addr↑,RL0;	4		N/A	
STAX wr	1	(wr)<—(A)		LDB mwr↑,RL0;	2		N/A	
STC	1	C<—1	C	SETFLG CY;	2		N/A	

TABLE D-1. 8080/8085 TRANZ CODE TRANSLATION SUMMARY (Cont.)

8080/8085 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH
SUB br	1	A<—(A)+(br)	C Z S P AC	SUBB RL0,mbr;	2	P	SUBB RL0,mbr;	
							ANDB RL0,RL0;	4
SUB M	1	A<—(A)+(M)	C Z S P AC	SUBB RL0,R3↑;	2	P	SUBB RL0,R3↑;	
							ANDB RL0,RL0;	4
SUI bv	2	A<—(A)-(bv)	C Z S P AC	SUBB RL0,bv;	4	P	SUBB RL0,bv;	
							ANDB RL0,RL0;	6
XCHG	1	(HL)<—>(DE)		EX R3,R2;	2		N/A	
XRA br	1	A<—(A)⊻(br)	0 Z S P 0	XORB RL0,mbr;	2	C	XORB RL0,mbr;	
							RESFLG CY;	4
XRA br (continued)						AC	XORB RL0,mbr;	
							LDCTLB RL4, FLAGS;	
							RESB RL4,3;	
							RESB RL4,2;	
							LDCTLB FLAGS,RL4;	10
XRA M	1	(A)<—(A)⊻(M)	0 Z S P 0	XORB RL0,R3↑;	2	C	XORB,RL0,R3↑;	
							RESFLG CY;	4
						AC	XORB RL0,R3↑;	
							LDCTLB RL4,FLAGS;	
							RESB RL4,3;	
							RESB RL4,2;	
							LDCTLB FLAGS,RL4;	10
XRI bv	2	A<—(A)⊻(bv)	0 Z S P 0	XORB RL0,bv;	4	C	XORB RL0,bv;	
							RESFLG CY;	6
						AC	XORB RL0,bv;	
							LDCTLB RL4,FLAGS;	
							RESB RL4,3;	
							RESB RL4,2;	
							LDCTLB FLAGS,RL4;	12
XTHL	1	(HL)<—((SP))		EX R3,R15↑;	2		N/A	

Note: 8085 Instructions RIM and SIM will not translate to AmZ8000 format since TRANZ does not support interrupt instructions. The error message UNTRANSLATABLE OPCODE ENCOUNTERED is issued.

APPENDIX E
Z80 CODE TRANSLATION SUMMARY

Table E summarizes the object code generated by TRANZ for each Z80 source code line. TRANZ allows the user to specify, through a series of prompts, Care or Don't Care conditions for the selected translation qualities. As shown the object code generated differs, depending on the setting of qualities. With some instructions, up to five lines of object code (14 bytes) can be generated to satisfy the Care-code settings specified by the user.

The table also defines the FCB flag bits affected by the instruction; i.e., bits modified or set to a specific value to reflect the results of the operation. A Ø or 1 entry in the flags-affected column indicates the constant value of the related bit following execution of the instruction; the flag code letter in the entry indicates that the flag is used to define the CPU condition following execution, and a blank indicates that the flag setting is unchanged.

The FCB flag bits and translation qualities are defined as:

C	=	Carry
Z	=	Zero
S	=	Sign
P/V	=	Parity/Overflow (same as AmZ8000 P/OV, similiar to 8080 P)
A/H	=	Auxiliary Carry (same as AmZ8000 H, Half Carry similiar to 8080 AC)
B	=	Byte Swap (word/byte reference order reversal)
K	=	Stack Byte Swap (word/byte stack reference order reversal)
F	=	Flag Bit ordering
D/N	=	Decimal Adjust (same as Z8000 DA flag)

Within the instructions, the following abbreviations and symbols are used:

br	=	byte register
bv	=	byte value
mbr	=	mapped byte register

mwr	= mapped word register
wr	= word register
wv	= word value
ir	= index register, e.g. Z80 IX or IY register.
mir	= mapped index register, i.e. Z8000 register (used to simulate the Z80 IX or IY register) R11 and R12 (respectively).
b	= a bit number, 0 through 7.
(X)	= contents of X
$(\overline{X})$	= complement of contents of X
((X))	= contents of contents of X. Contents of designated location in inner brackets are interpreted as address to a second location that contains the 8-bit operand.
↑X	= address of X
X↑	= item referenced by the address of X
<—	= receives
<—>	= is exchanged with
/\	= logical AND
V	= logical OR
$\forall$	= logical EXCLUSIVE OR

Note: Unless otherwise specified in the table, translated instructions are modified for AmZ8000 compability. In some cases, the modified inexact object code is flagged with a message to that effect. Where only the length of the code is changed, a change message is not issued. Unless flagged, the resultant object code is functionally identical to the source code.

+	= indicates that the preceeding flag is not supported for the instruction translation e.g. no translation fix up is provided.

TABLE E-1. Z80 CODE TRANSLATION SUMMARY

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH
ADC A,(HL)	1	A<—A+src+CY	C Z S V 0 H	LDB RL4,R3↑; ADCB RL0,RL4;	4		N/A	

ADC A,(ir+bv)	3		C Z S V 0 H	LDB RL4,mir↑(bv);
				ADCB RL0,RL4;
					6			

ADC A, br	1		C Z S V 0 H	ADCB RL0,mbr;	2			

ADC A, bv	2		C Z S V 0 H	LDB RL4,bv;
				ADCB RL0,RL4;
					6			

ADC HL,wr	2		C Z S V 0 H^{+}	ADC R3,mwr;	2			
ADD A, (HL)	1	A<—A+src	C Z S V 0 H	ADDB RL0,R3↑;	2			
ADD A, (ir+bv)	3		C Z S V 0 H	ADDB RL0,mir↑(bv);	4			
ADD A,br	1		C Z S V 0 H	ADDB RL0,mbr;	2			
ADD A,bv	2		C Z S V 0 H	ADDB RL0,bv;	4			

ADD wr,wr	1-2		C 0 H^{+}	ADD mwr,mwr;	2	Z
						or
						S
						or
						P
							LDCTLB RL4,FLAGS
							ADD mwr,mwr;
							JR NC,(^$+6)^;
							SETB RL4,7;
							JR (^$+4)^;
							RESB RL4,7;
							LDCTLB FLAGS,RL4;
								14

AND (HL)	1	A<—A/\src	0 Z S P 0 1^{+}	ANDB RL0,R3↑;	2	C	ANDB RL0,R3↑; RESFLG C;	4
AND (ir+bv)	3		0 Z S P 0 1^{+}	ANDB RL0,mir↑(bv);	4	C	ANDB RL0,mir↑(bv) RESFLG C;	6

AND br	1		0 Z S P 0 1^{+}	ANDB RL0,mbr;	2	C	ANDB RL0,mbr;
							RESFLG C;
								4

AND bv	2		0 Z S P 0 1^{+}	ANDB RL0,bv;	4	C	ANDB RL0,bv;
							RESFLG C;	

BIT b,(HL)	2	Z<—$\overline{\text{(src)}}$b	Z ? ? 0 1^{+}	BITB R3↑,b;	2			6

TABLE E-1. Z80 CODE TRANSLATION SUMMARY (Cont.)

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH
BIT b,(ir+bv)	4		Z ? ? 0 1^+	BITB mir↑(bv),b;	4			
BIT b,br	2		Z ? ? 0 1^+	BITB mbr,b;	2			

CALL cond,addr	3	if cond.
		then CALL addr
				JR not(cond),(↑$+6)↑;
				CALL addr;
					6	K	JR not(cond),(^$+14)^;
							LD R5,addr;
							EXB RH5,RL5;
							PUSH R15↑,R5;
							JP addr;
								14

CALL addr	3	SP<—SP-2
		(SP)<—PC
		PC<—addr
				CALL addr;	4	K	LD R5,addr;
							EXB RH5,RL5;
							PUSH R15↑,R5;
							JP addr;
								12

CCF	1	C<—$\overline{C}$	C 0 H^+	COMFLG CY;	2			
CP (HL)	1	A-src	C Z S V 1 H^+	CPB RL0,R3↑;	2			
CP (ir+bv)	3		C Z S V 1 H^+	CPB RL0,mir↑(bv);	4			
CP br	1		C Z S V 1 H^+	CPB RL0,mbr;	2			
CP bv	2		C Z S V 1 H^+	CPB RL0,bv;	4			

CPD	2	A-(HL)
		HL<—HL-1
		BC<—BC-1
			Z S V 1 H^+	CPDB RL0,R3↑,R1,ZR;	4	S	CPDB RL0,R3↑,R1,ZR;
							LDCTLB RL4,FLAGS;
							CPB RL0,R3↑(1);
							JR PL,(↑$+6)↑;
							SETB RL4,5;
							JR (↑$+4)↑;
							RESB RL4,5;
							LDCTLB FLAGS,RL4;
								20

CPDR	2	DO: A-(HL)
		HL<—HL-1
		BC<—BC-1
		Until:A=(HL) or
		BC=0
			Z S V 1 H^+	CPDRB RL0,R3↑,R1,ZR;	4	S	CPDRB RL0,R3↑,R1,ZR;
							LDCTLB RL4,FLAGS;
							CPB RL0,R3↑(1);
							JR PL,(↑$+6)↑;
							SETB RL4,5;
							JR (↑$+4)↑;
							RESB RL4,5;
							LDCTLB FLAGS,RL4;
								20

TABLE E-1. Z80 CODE TRANSLATION SUMMARY (Cont.)

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH

CPI	2	A-(HL)
		HL<—HL+1
		BC<—BC-1
			Z S V 1 H^+	CPIB RL0,R3↑,R1,ZR;	4	S	CPIB RL0,R3↑,R1,ZR;
							LDCTLB RL4,FLAGS;
							CPB RL0,R3↑(-1);
							JR PL,(↑$+6)↑;
							SETB RL4,5;
							JR (↑$+4)↑;
							RESB RL4,5;
							LDCTLB FLAGS,RL4;
								20

CPIR	2	DO: A-(HL)
		HL<—HL+1
		BC<—BC-1
		Until:A=(HL) or BC=0
			Z S V 1 H^+	CPIRB RL0,R3↑,R1,ZR;	4	S	CPIRB RL0,R3↑,R1,ZR;
							LDCTLB RL4,FLAGS;
							CPB RL0,R3↑(-1);
							JR PL,(↑$+6)↑;
							SETB RL4,5;
							JR (↑$+4)↑;
							RESB RL4,5;
							LDCTLB FLAGS,RL4;
								20

CPL	1	A<—$\overline{A}$	1 1^+	COMB RL0;		C or Z or S	LDCTLB RL4,FLAGS;
							COMB RL0;
							LDCTLB FLAGS,RL4;
								6

DAA	1	A<—2BCD(A)	C Z S P H	DAB RL0;	2	P	DAB RL0;
							ANDB RL0,RL0;
								4

DEC (HL)	1	dst<—dst-1	Z S P 1 H	DECB R↑,1;	2	A	LDCTLB RL4,FLAGS;
							SUBB R3↑,1;
							RESFLG CY;
							LDCTLB RL5,FLAGS;
							BITB RL4,7;
							JR ZR,(↑$+4)↑;
							SETB RL5,7;
							LDCTLB FLAGS,RL5;
								18

DEC (ir+bv)	3		Z S V 1 H	DECB mir↑(bv),1;	4	A	LDCTLB RL4,FLAGS;
							SUBB mir↑(bv),1;
							RESFLG CY;
							LDCTLB RL5,FLAGS;
							BITB RL4,7;
							JR ZR,(↑$+4)↑;
							SETB RL5,7;
							LDCTLB FLAGS,RL5;
								20

TABLE E-1. Z80 CODE TRANSLATION SUMMARY (Cont.)

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH

DEC br	1		Z S V 1 H	DECB mbr,1;	2	A	LDCTLB RL4,FLAGS;
							SUBB mbr,1;
							RESFLG CY;
							LDCTLB RL5,FLAGS;
							BITB RL4,7;
							JR ZR,(↑$+4)↑;
							SETB RL5,7;
							LDCTLB FLAGS,RL5;
								18

DEC wr	1-2			DEC mwr,1;	2	Z
						or
						S
						or
						P
							LDCTLB RL4,FLAGS;
							DEC mwr,1;
							LDCTLB FLAGS,RL4;
								6

DI	1	disable interrupts		DI VI,NVI;	2			
DJNZ addr	2	B<—B-1 if B≠0 then JP addr		DBJNZ RH1,addr;	2			
EI	1	enable interrupts		EI VI,NVI;	2			
EX (SP),wr	1-2	(SP)<—wr		EX R15↑,mwr;	2			

EX AF,AF′		AF<—>AF′		LDCTLB RH0,FLAGS;
				EX R0,R7;
				LDCTLB FLAGS,RH0;
					6	F	CALL @@@F2;
							EX R0,R7↑;
							CALL @@@F1;
								10

EXX	1	BC<—>BC′
		DE<—>DE′
		HL<—>HL′
				EX R1,R8;
				EX R2,R9;
				EX R3,R10;
					6			

HALT	1	halt		HALT;	2			
IM n	2	Set interrupt mode		*				
IN A,(bv)	2	A<—port bv		INB RL0,bv;	4		**	

IN br,(C)	2	br<—port (C)		LDB RL4,RL1;
				LDB RH4,0;
				INB mbr,R4;
					6		**	

* Untranslatable instruction. An error message is issued.
** Translation is not exact for hardware. Program need not be modified. A warning is issued.

TABLE E-1. Z80 CODE TRANSLATION SUMMARY (Cont.)

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH

INC (HL)	1	dst<—dst+1	Z S V 0 H	INCB R3↑,1;	2	A
						or
						D
							LDCTLB RL4,FLAGS;
							ADDB R3↑,1;
							RESFLG CY;
							LDCTLB RL5,FLAGS;
							BITB RL4,7;
							JR ZR,(↑$+4)↑;
							SETB RL5,7;
							LDCTLB FLAGS,RL5;
								18

INC (ir+bv)	3		Z S V 0 H	INCB mir↑(bv),1;	4	A
						or
						D
							LDCTLB RL4,FLAGS;
							ADDB mir↑(bv),1;
							RESFLG CY;
							LDCTLB RL5,FLAGS;
							BITB RL4,7;
							JR ZR,(↑$+4)↑;
							SETB RL5,7;
							LDCTLB FLAGS,RL5;
								20

INC br	1		Z S V 0 H	INCB mbr,1;	2	A
						or
						D
							LDCTLB RL4,FLAGS;
							ADDB RL0 1;
							RESFLG CY;
							LDCTLB RL5,FLAGS;
							BITB RL4,7;
							JR ZR,(↑$+4)↑;
							SETB RL5,7;
							LDCTLB FLAGS,RL5;
								18

INC wr	1-2			INC mwr,1;	2	Z
						or
						S
						or
						P
							LDCTLB RL4,FLAGS;
							INC mwr,1;
							LDCTLB FLAGS,RL4;
								6

IND	2	(HL)<—(C)
		B<—B-1
		HL<—HL-1
			Z ? ? 1^{+}?	LDB RL4,RH1;
				LDB RH4,0;
				LDB RH1,0;
				INDB R3↑,R1,R4;
				LDB RH1,RL4;
				JR OV,(↑$+6)↑;
				RESFLG ZR;
				JR (↑$+4)↑;
				SETFLG ZR;
					18		**	

TABLE E-1. Z80 CODE TRANSLATION SUMMARY (Cont.)

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH

INDR	2	DO: (HL)<—(C)
		B<—B-1
		HL<—HL-1
		Until:B=0
			1 ? ? 1^{+}?	LDB RL4,RH1;
				LDB RH4,0;
				LDB RH1,0;
				INDRB R3↑,R1,R4;
				LDB RH1,RL4;
				JR OV,(↑$+6)↑;
				RESFLG ZR;
				JR (↑$+4)↑;
				SETFLG ZR;
					18	**		

INI	2	(HL)<—(C)
		B<—B-1
		HL<—HL+1
			Z ? ? 1^{+}?	LDB RL4,RH1;
				LDB RH4,0;
				LDB RH1,0;
				INIB R3↑,R1,R4;
				LDB RH1,RL4;
				JR OV,(↑$+6)↑;
				RESFLG ZR;
				JR (↑$+4)↑;
				SETFLG ZR;
					18	**		

INIR	2	DO: (HL)—(C)
		B<—B-1
		HL<—HL+1
		Until:B=0
			1 ? ? 1^{+}?	LDB RL4,RH1;
				LDB RH4,0;
				LDB RH1,0;
				INIRB R3↑,R1,R4;
				LDB RH1,RL4;
				JR OV,(↑$+6)↑;
				RESFLG ZR;
				JR (↑$+4)↑;
				SETFLG ZR;
					18		**	

JP (wr)	2	PC<—(wr)		JP wr↑;	2			
JP addr	3	PC<—addr		JP addr;	4			
JP cond,addr	3	If: cond then: JP addr		JP cond,addr;	4			
JR addr	2	PC<—addr		JR addr;	2			
JR cond,addr	2	If: cond then: JR addr		JR cond,addr;	2			

TABLE E-1. Z80 CODE TRANSLATION SUMMARY (Cont.)

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH
LD (wr),br	1	(wr)<—br		LDB mwr↑,mbr;	2			
LD (HL),bv	2	(HL)<—bv		LDB R3↑,bv;	4			
LD (ir+bv),br	3	(ir+bv)<—br		LDB mir↑(bv),br;	4			
LD (ir+bv),bv	4	(ir+bv)<—bv		LDB mir↑(bv),bv;	6			
LD (wv),A	3	(wv)<—A		LDB (wv)↑,RL0;	4			
LD (wv),wr	3-4	(wv)<—wr		LD (wv)↑,mwr;	4	B	EXB RHmwr,RLmwr; LD (wv)↑,mwr; EXB RHmwr,RLmwr;	8
LD br,(wr)	1	br<—(wr)		LDB mbr,mwr↑;	2			
LD br,(ir+bv)	3	br<—(ir+bv)		LDB mbr,mir↑(bv);	4			
LD br,br	1	br<—br		LDB mbr,mbr;	2			
LD br,bv	2	br<—bv		LDB mbr,bv;	2			
LD A,(wv)	3	A<—(wv)		LDB RL0,(wv)↑;	4			
LD wr,(wv)	3-4	wr<—(wv)		LD mwr,(wv)↑;	4	B	LD mwr,(wv)↑; EXB RH mwr, RL mwr;	6
LD wr,wr	1-2	wr<—wr		LD mwr,mwr;	2			
LD wr,wv	3	wr<—wv		LD mwr,wv;	4			
LD A,I	2	A<—interrupt		*				
LD I,A	2	interrupt<—A		*				
LD R,A	2	refresh<—A		LDB RH0,0; LDCTL REFRESH,R0;	4		**	
LDD	2	(DE)<—(HL) DE<—DE-1 HL<—HL-1 BC<—BC-1	V $0^{+}0^{+}$	LDDB R2↑,R3↑,R1;	4	P	LDDB R2↑,R3↑,R1; COMFLG PY;	6

*Untranslatable instruction. An error message is issued.

TABLE E-1. Z80 CODE TRANSLATION SUMMARY (Cont.)

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH

LDDR	2	DO· (DE)<—(HL)
		DE<—DE-1
		HL<—HL-1
		BC<—BC-1
		Until:BC=0
			0 $0^{+}0^{+}$	LDDRB R2↑,R3↑,R1;	4	P	LDDRB R2↑,R3↑,R1;
							COMFLG PY;
								6

LDI	2	(DE)<—(HL)
		DE<—DE+1
		HL<—HL+1
		BC<—BC+1
			V $0^{+}0^{+}$	LDIB R2↑,R3↑,R1;	4	P	LDIB R2↑,R3↑,R1;
							COMFLG PY;
								6

LDIR	2	DO: (DE)<—(HL)
		DE<—DE+1
		HL<—HL+1
		BC<—BC-1
		Until:BC=0
			0 $0^{+}0^{+}$	LDIRB R2↑,R3↑,R1;	4	P	LDIRB R2↑,R3↑,R1;
							COMFLG PY;
								6

NEG	2	A<—0-A	C Z S V 1 H	NEGB RL0;	2			
NOP	1	No operation		NOP;	2			
OR (HL)	1	A<—A V src	0 Z S P $0^{+}0^{+}$	ORB RL0,R3↑;	2	C	ORB RL0,R3↑; RESFLG CY;	6

OR (ir+bv)	3		0 Z S P $0^{+}0^{+}$	ORB RL0,mir↑(bv);	4	C	ORB RL0,mir↑(bv);
							RESFLG CY;
								6

OR br	1		0 Z S P $0^{+}0^{+}$	ORB RL0,mbr;	2	C	ORB RL0,mbr;
							RESFLG CY;
								6

OR bv	2		0 Z S P $0^{+}0^{+}$	ORB RL0,bv;	4	C	ORB RL0,bv;
							RESFLG CY;
								6

OTDR	2	DO: (C)<—(HL)
		B<—B-1
		HL<—HL-1
		Until:B=0
			1 ? ? 1^{+}?	LDB RL4,RH1;
				LDB RH4,0;
				LDB RH1,0;
				OTDRB R1,R3↑,R4;
				LDB RH1,RL4;
				JR OV,(↑$+6)↑;
				RESFLG ZR;
				JR (↑$+4)↑;
				SETFLG ZR;
					18		**	

TABLE E-1. Z80 CODE TRANSLATION SUMMARY (Cont.)

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH

OTIR	2	DO: (C)<—(HL)
		B<—B-1
		HL<—HL+1
		Until:B=0
			1 ? ? 1^{+}?	LDB RL4,RH1;
				LDB RH4,0;
				LDB RH1,0;
				OTIRB R1,R3↑,R4;
				LDB RH1,RL4;
				JR OV,(↑$+6)↑;
				RESFLG ZR;
				JR (↑$+4)↑;
				SETFLG ZR;
					18		**	

OUT (c),br	2	port (C)<—br		LDB RL4,RL1;
				LDB RH4,0;
				OUTB mbr,R4;
					6		**	

OUT (bv),A	2	port bv<—A		OUTB RL0,bv;	4		**	

OUTD	2	(C)<—(HL)
		B<—B-1
		HL<—HL-1
			Z ? ? 1^{+}?	LDB RL4,RH1;
				LDB RH4,0;
				LDB RH1,0;
				OUTD R1,R3↑,R4;
				LDB RH1,RL4;
				JR OV,(↑$+6)↑;
				RESFLG ZR;
				JR (↑$+4)↑;
				SETFLG ZR;
					18		**	

OUTI	2	(C)<—(HL)
		B<—B-1
		HL<—HL+1
			Z ? ? 1^{+}?	LDB RL4,RH1;
				LDB RH4,0;
				LDB RH1,0;
				LDB RH1,RL4;
				JR OV,(↑$+6)↑;
				RESFLG ZR;
				JR (↑$+4)↑;
				SETFLG ZR;
					18		**	

POP AF	1	PSW<—(SP)
		SP<—SP+2
			C Z S P N H	POP R0,R15↑;
				LDCTLB FLAGS,RH0;
					4	K	POP R0,R15↑;
							EXB RL0,RH0;
							LDCTLB FLAGS,RH1;
								6

						F	POP R0,R15↑
							CALL @@@F1;
							LDCTLB FLAGS,RH1;
								8

TABLE E-1. Z80 CODE TRANSLATION SUMMARY (Cont.)

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH
POP wr	1-2	wr<—(SP) SP<—SP+2		POP mwr,R15↑;	2	K	POP mwr,R15↑; EX RHmwr,RLmwr;	4

PUSH AF	1	SP<—SP-2
		(SP)<—PSW
				LDCTLB RH0,FLAGS;
				PUSH R15↑,R0;
					4	K	LDCTLB RH1,FLAGS;
							EXB RH0,RL0;
							PUSH R15↑,R0;
							EXB RH0,RL0;
								8

					2	F	CALL @@@F2;
							PUSH R15↑,R0;
								6

PUSH wr	1-2	SP<—SP-2
		(SP)<—wr
				PUSH R15↑,mwr;	2	K	EXB RHmwr,RLmwr;
							PUSH R15↑,mwr;
							EXB RHmwr,RLmwr;
								6

RES b,(HL)	2	(HL)$_b$<—0		RESB R3↑,b;	2			
RES b,(ir+bv)	4	(ir+bv)$_b$<—0		RESB mir↑(bv);	4			
RES b,br	2	br$_b$<—0		RESB mbr,b;	2			
RET	1	PC<—(SP) SP<—SP+2		RET	2	K	POP R5,R15↑; EXB RH5,RL5; JP R5↑;	6
RET cond	1	if: cond then: RET		RET cond;	2	K	JR NOT cond,(↑$+8)↑; POP R5,R15↑; EXB RH5,RL5; JR R5↑;	8
RETI	2	Return from interrupt		(see RET)			***	
RETN	2	Return from NMI		(see RET)			***	

RL (HL)	2	CY<—b$_7$
		b$_{1-7}$<—b$_{0-6}$
		b$_0$<—CY
			C Z S P $0^{+}0^{+}$	LDB RL4,R3↑
				RLCB RL4,1;
				LDB R3↑,RL4;
					6	P	LDB RL4,R3↑;
							RLCB RL4,1;
							LDB R3↑,RL4;
							ANDB RL4,RL4;
								8

RL (ir+bv)	4		C Z S P $0^{+}0^{+}$	LDB RL4,mir↑(bv);
				RLCB RL4,1;
				LDB mir↑(bv),RL4;
					10	P	LDB RL4,mir↑(bv);
							RLCB RL4,1;
							LDB mir↑(bv),RL4;
							ANDB RL4,RL4;
								12

*** Exact translation is not possible - the return instruction is translated. A warning message is issued.

TABLE E-1. Z80 CODE TRANSLATION SUMMARY (Cont.)

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH
RL br	2		C Z S P $0^{+}0^{+}$	RLCB mbr,1;	2	P	RLCB mbr,1; ANDB mbr,mbr,	4

RLA	1		C $0^{+}0^{+}$	RLCB RL0,1;	2	Z
						or
						S
						or
						P
							LDCTLB RL4,FLAGS;
							RLCB RL0,1;
							JR NC,(↑$+6)↑;
							SETB RL4,7;
							JR (↑$+4)↑;
							RESB RL4,7;
							LDCTLB FLAGS,RL4;
								14

RLC (HL)	2	$CY \leftarrow b_7$
		$b_{1-7} \leftarrow b_{0-6}$
		$b_0 \leftarrow b_7$
			C Z S P $0^{+}0^{+}$	LDB RL4,R3↑;
				RLB RL4,1;
				LDB R3↑,RL4;
					6	P	LDB RL4,R3↑;
							RLB RL4,1;
							LDB R3↑,RL4;
							ANDB RL4,RL4;
								8

RLC (ir+bv)	4		C Z S P $0^{+}0^{+}$	LDB RL4,mir↑(bv);
				RLB RL4,1;
				LDB mir↑(bv),RL4;
					10	P	LDB RL4,mir↑(bv)
							RLB RL4,1;
							LDB mir↑(bv),RL4;
							ANDB RL4,RL4;
								12

RLC br	2		C Z S P $0^{+}0^{+}$	RLB mbr,1;	2	P	RLB mbr,mbr;
							ANDB mbr,mbr;
								4

RLCA	1		C $0^{+}0^{+}$	RLB RL0,1;	2	Z
						or
						S
						or
						P
							LDCTLB RL4,FLAGS;
							RLB RL0,1;
							JR NC,(↑$+6)↑;
							SETB RL4,7;
							JR (↑$+4)↑;
							RESB RL4,7;
							LDCTLB FLAGS,RL4;	

RLD	2	$A_{0-3} \leftarrow (HL)_{4-7}$
		$(HL)_{4-7} \leftarrow (HL)_{0-3}$
		$(HL)_{0-3} \leftarrow A_{0-3}$
			Z S P $0^{+}0^{+}$	LDB RL4,R3↑;
				RLDB RL0,RL4;
				LDB R3↑,RL4;
					6	P	LDB RL4,R3↑;
							RLDB RL0,RL4;
							LDB R3↑,RL4;
							ANDB RL0,RL0;
								8

RR(HL)	2	$b_7 \leftarrow CY$
		$b_{0-6} \leftarrow b_{1-7}$
		$CY \leftarrow b_0$
			C Z S P $0^{+}0^{+}$	LDB RL4,R3↑;
				RRCB RL4,1;
				LDB R3↑,RL4;
					6	P	LDB RL4,R3↑;
							RRCB RL4,1;
							LDB R3↑,RL4;
							ANDB RL4,RL4;
								8

TABLE E-1. Z80 CODE TRANSLATION SUMMARY (Cont.)

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH

RR (ir+bv)	4		C Z S P 0^+0^+	LDB RL4,mir↑(bv);
				RRCB RL4,1;
				LDB mir↑(bv),RL4;
					10	P	LDB RL4,mir↑(bv);
							RRCB RL4,1;
							LDB mir↑(bv),RL4;
							ANDB RL4,RL4;
								12

RR br	2		C Z S P 0^+0^+	RRCB mbr,1;	2	P	RRCB mbr,1; ANDB mbr,mbr,	4

RRA	1		C 0^+0^+	RRCB RL0,1;	2	Z
						or
						S
						or
						P
							LDCTLB RL4,FLAGS;
							RRCB RL0,1;
							JR NC,(↑$+6)↑;
							SETB RL4,7;
							JR (↑$+4)↑;	

RRC (HL)	2	CY←b_0
		b_{0-6}←b_{1-7}
		b_7←b_0
			C Z S P 0^+0^+	LDB RL4,R3↑;
				RRB RL4,1;
				LDB R3↑,RL4;
					6	P	LDB RL4,R3↑;
							RRB RL4,1;
							LDB R3↑,RL4;
							ANDB RL4,RL4;
								8

RRC (ir+bv)	4		C Z S P 0^+0^+	LDB RL4,mir↑(bv);
				RRB RL4,1;
				LDB mir↑(bv),RL4;
					10	P	LDB RL4,mir↑(bv);
							RRB RL4,1;
							LDB mir↑(bv),RL4;
							ANDB RL4,RL4;
								12

RRC br	2		C Z S P 0^+0^+	RRB mbr,1;	2	P	RRB mbr,mbr;
							ANDB mbr,mbr;
								4

RRCA	1		C 0^+0^+	RRB RL0,1;	2	Z
						or
						S
						or
						P
							LDCTLB RL4,FLAGS;
							RRB RL0,1;
							JR NC,(↑$+6)↑;
							SETB RL4,7;
							JR (↑$+4)↑;
							RESB RL4,7;
							LDCTLB FLAGS,RL4;	

RRD	2	$(HL)_{4-7}$←A_{0-3}
		$(HL)_{0-3}$←$(HL)_{4-7}$
		A_{0-3}←$(HL)_{0-3}$
			Z S P 0^+0^+	LDB RL4,R3↑;
				RRDB RL0,RL4;
				LDB R3↑,RL4;
					6	P	LDB RL4,R3↑;
							RRDB RL0,RL4;
							LDB R3↑,RL4;
							ANDB RL0,RL0;
								8

RST addr	1	CALL addr		CALL addr;	4	K	LD R5,addr;
							EXB RH5 RL5;
							PUSH R15↑,R5;
							JP addr;
								12

TABLE E-1. Z80 CODE TRANSLATION SUMMARY (Cont.)

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH
SBC A,(HL)	1	$A \leftarrow A\text{-src-}CY$	C Z S V 1 H	LDB RL4,R3↑; SBCB RL0,RL4;	4			

SBC A,(ir+bv)	3		C Z S V 1 H	LDB RL4,mir↑(bv);
				SBCB RL0,RL4;
					6			

SBC A,br	1		C Z S V 1 H	SBCB RL0,mbr;	2			
SBC A,bv	2		C Z S V 1 H	LDB RL4,bv, SBCB RL0,RL4;	4			
SBC HL,wr	2		C Z S V 1 H	SBC R3,mwr;	2			
SCF	1	$CY \leftarrow 1$	1	SETFLG CY;	2			
SET b,(HL)	2	$(HL)_b \leftarrow 1$		SETB R3↑,b;	2			
SET b,(ir+bv)	4	$(ir+bv)_b \leftarrow 1$		SETB mir↑(bv),b;	4			
SET b,br	2	$br_b \leftarrow 1$		SETB mbr,b;	2			
SLA (HL)	2	$CY \leftarrow b_7$ $b_{1-7} \leftarrow b_{0-6}$ $b_0 \leftarrow 0$	C Z S P 0^+0^+	LDB RL4,R3↑; SLAB RL4,1; LDB R3↑,RL4,	8			

SLA (ir+bv)	4		C Z S P 0^+0^+	LDB RL4,mir↑(bv);
				SLAB RL4,1;
				LDB mir↑(bv),RL4;
					12			

SLA br	2		C Z S P 0^+0^+	SLAB mbr;	4			
SRA (HL)	2	$CY \leftarrow b_0$ $b_{0-6} \leftarrow b_{1-7}$	C Z S P 0^+0^+	LDB RL4,R3↑; SRAB RL4,1; LDB R3↑,RL4,	8			

SRA (ir+bv)	4		C Z S P 0^+0^+	LDB RL4,mir↑(bv);
				SRAB RL4,1;
				LDB mir↑(bv),RL4;
					12			

SRA br	2		C Z S P 0^+0^+	SRAB mbr;	4			
SRL (HL)	2	$CY \leftarrow b_0$ $b_{0-6} \leftarrow b_{1-7}$ $b_7 \leftarrow 0$	C Z S P 0^+0^+	LDB RL4,R3↑; SRLB RL4,1; LDB R3↑,RL4,	8			

TABLE E-1. Z80 CODE TRANSLATION SUMMARY (Cont.)

Z80 INSTRUCTION	BYTE LENGTH	OPERATION	FLAGS AFFECTED	AmZ8000 DON'T CARE CODE GENERATED	TOTAL BYTE LENGTH	CARE QUAL. SET	AmZ8000 CARE CODE GENERATED	TOTAL BYTE LENGTH

SRL (ir+bv)	4		C Z S P 0^+0^+	LDB RL4,mir↑(bv);
				SRLB RL4,1;
				LDB mir↑(bv),RL4;
					12			

SRL br	2		C Z S P 0^+0^+	SRLB mbr;	4			
SUB (HL)	1	A<—A-src	C Z S V 1 H	SUBB RL0,R3↑;	2			
SUB (ir+bv)	3		C Z S V 1 H	SUBB RL0,mir↑(bv);	4			
SUB br	1		C Z S V 1 H	SUBB RL0,mbr;	2			4
SUB bv	2		C Z S V 1 H	SUBB RL0,bv;	4			6
XOR (HL)	1	A<—A ⊻ src	0 Z S P 0^+0^+	XORB RL0,R3↑;	2	C	XORB RL0,R3↑; RESFLG CY;	

XOR (ir+bv)	3		0 Z S P 0^+0^+	XORB RL0,mir↑(bv);	4	C	XORB RL0,mir↑(bv);
							RESFLG CY;
								4

XOR br	1		0 Z S P 0^+0^+	XORB RL0,mbr;	2	C	XORB RL0,mbr;
							RESFLG CY;
								6

XOR bv	2		0 Z S P 0^+0^+	XORB RL0,bv;	4	C	XORB RL0,bv;
							RESFLG CY;	

COMMENT SHEET

Address comments to:

Advanced Micro Computers
Publications Department
3340 Scott Boulevard
Santa Clara, CA 95051

TITLE: TRANZ TRANSLATOR USER'S MANUAL
PUBLICATION NO. 00680118C

COMMENTS: (Describe errors, suggested additions or deletions, and include page numbers, etc.)

From: Name: ______________________ Position: ______________

Company: ______________________

Address: ______________________

Advanced Micro Computers

A subsidiary of
Advanced Micro Devices
3340 Scott Boulevard
Santa Clara,
California 95051
(408) 988-7777
TELEX: 171 142