Description		Mnemonic	Operands	Addressing Modes	Status Bits	Modifies	Corrupts
Stop execution	0	**STOP**					
Return from CALL	0	**RET**				SP	
Return from trap	0	**RETTR**					
Move SP to A	0	**MOVSPA**				A	
Move NZVC flags to $A_{12:15}$	0	**MOVFLGA**				A	
Move $A_{12:15}$ to NZVC flags	0	**MOVAFLG**					
Bitwise invert r	0	**NOTr**			NZ	r	
Negate r	0	**NEGr**			NZV	r	
Arithmetic shift left r	0	**ASLr**			NZVC	r	
Arithmetic shift right r	0	**ASRr**			NZC	r	
Rotate left r	0	**ROLr**			C	r	
Rotate right r	0	**RORr**			C	r	
Branch unconditional	0	**BR**	label	*i,x*			
Branch if less than or equal	0	**BRLE**	label	*i,x*			
Branch if less than	0	**BRLT**	label	*i,x*			
Branch if equal to	0	**BREQ**	label	*i,x*			
Branch if not equal to	0	**BRNE**	label	*i,x*			
Branch if greater than or	0	**BRGE**	label	*i,x*			
Branch if greater than	0	**BRGT**	label	*i,x*			
Branch if oVerflow	0	**BRV**	label	*i,x*			
Branch if Carry	0	**BRC**	label	*i,x*			
Call subprogram	0	**CALL**	label	*i,x*		SP	
Unary no operation trap	0	**NOPn**					
Nonunary no operation trap	0	**NOP**		*i*			
Decimal input trap	0	**DECI**	m,ams	*d,n,s,sf,x,sx,sfx*	NZV	m	
Decimal output trap	0	**DECO**	m,ams	*i,d,n,s,sf,x,sx,sfx*			
Hexadecimal output trap	0	**HEXO**	m,ams	*i,d,n,s,sf,x,sx,sfx*			
String output trap	0	**STRO**	m,ams	*d,n,s,sf,x*			
Add to stack pointer	0	**ADDSP**	m,ams	*i,d,n,s,sf,x,sx,sfx*	NZVC	SP	
Subtract from stack pointer	0	**SUBSP**	m,ams	*i,d,n,s,sf,x,sx,sfx*	NZVC	SP	
Add to r	0	**ADDr**	m,ams	*i,d,n,s,sf,x,sx,sfx*	NZVC	r	
Subtract from r	0	**SUBr**	m,ams	*i,d,n,s,sf,x,sx,sfx*	NZVC	r	
Bitwise AND to r	0	**ANDr**	m,ams	*i,d,n,s,sf,x,sx,sfx*	NZ	r	
Bitwise OR to r	0	**Orr**	m,ams	*i,d,n,s,sf,x,sx,sfx*	NZ	r	
Compare $r_{0:15}$ to word	0	**CPWr**	m,ams	*i,d,n,s,sf,x,sx,sfx*	NZVC		
Compare $r_{8:15}$ to byte	0	**CPBr**	m,ams	*i,d,n,s,sf,x,sx,sfx*	NZVC		
Load word $r_{0:15}$ from memory	0	**LDWr**	m,ams	*i,d,n,s,sf,x,sx,sfx*	NZ	$r_{0:15}$	
Load byte $r_{8:15}$ from memory	0	**LDBr**	m,ams	*i,d,n,s,sf,x,sx,sfx*	NZ	$r_{8:15}$	
Store word $r_{0:15}$ to memory	0	**STWr**	m,ams	*d,n,s,sf,x,sx,sfx*		m	
Store byte $r_{8:15}$ to memory	0	**STBr**	m,ams	*d,n,s,sf,x,sx,sfx*		m	

Description		Directive					
The address of a symbol	0	**.ADDRSS**	label				
Padding to align at boundary	0	**.ALIGN**	number				
A string of ASCII bytes	0	**.ASCII**	string				
A block of bytes	0	**.BLOCK**	number				
Initiate ROM burn	0	**.BURN**					
A byte value	0	**.BYTE**	number				
The sentinel for the	0	**.END**					
Equate a symbol to a	0	**.EQUATE**	number				
number	0	**.WORD**	number				

Description		Directive					
Include file at this point	1	.INCLUDE	*"fid"*				
Append file at the end	1	.APPEND	*"fid"*				
Declare with global scope	1	.GLOBAL	label				
Begin a macro definition	1	.MACRO					
End a macro definition	1	.MACROEND					
Description		**Mnemonic**	**Operands**	**Addressing**	**Status**	**Modifies**	**Corrupts**
Branch if N	2	BRN	label				
Branch if not N	2	BRNN	label				
Branch if Z	2	BRZ	label				
Branch if not Z	2	BRNZ	label				
Branch if not oVerflow	2	BRNV	label				
Branch if not Carry	2	BRNC	label				
Clear word $r_{0:15}$	2	CLRWr			NZ	r	
Clear byte $r_{8:15}$	2	CLRBr			NZ	$r_{8:15}$	
Clear word memory	2	CLRW	m,ams			m	NZVC
Clear byte memory	2	CLRB	m,ams			m	NZVC
Increment r	2	INCr			NZVC	r	
Decrement r	2	DECr			NZVC	r	
Increment memory	2	INC	m,ams			m	NZVC
Decrement memory	2	DEC	m,ams			m	NZVC
Bitwise invert memory	2	NOT	m,ams			m	NZVC
Negate memory	2	NEG	m,ams			m	NZVC
Add; m ← m + addend	2	ADD	m,ams,addend,ams2			m	NZVC
Subtract; m ← m - subend	2	SUB	m,ams,subend,ams2			m	NZVC
Move; destin ← source	2	MOVE	source,ams,destin,ams2			destin	NZVC
Compare word memory	2	CPW	m1,ams,m2,ams2		NZVC		A
Compare byte memory	2	CPB	m1,ams,m2,ams2		NZVC		A
Test (compare to zero) $r_{0:15}$	2	TSTWr			NZVC		
Test (compare to zero) $r_{8:15}$	2	TSTBr			NZVC		
Test word memory	2	TSTW	m,ams		NZVC		A
Test byte memory	2	TSTB	m,ams		NZVC		A
PUSH r	2	PUSHr				SP	NZVC
POP r	2	POPr				SP,r	NZVC
PUSH memory	2	PUSH	m,ams			SP	NZVC
POP memory	2	POP	m,ams			SP,m	NZVC
Character Input	2	CHARI	m,ams				
Character Output	2	CHARO	m,ams				
String Input[1] to memory	2	STRI[1]	m,ams			m[2],A[2]	NZVC
Decimal Input r	2	DECIr				r	NZVC
Decimal Output r	2	DECOr					
Binary Output r	2	BINOr					NZVC
Binary Output memory	2	BINO	m,ams				NZVC
Hexadecimal Output r	2	HEXOr					NZVC
Dump Stack (top portion)	2	DUMPS	m,ams				NZVC
Statically save r	2	SAVEr[3]					
Statically save A,X	2	SAVE[4]					
Restore saved r	2	RESTOREr[3]				r	NZVC
Restore saved A,X	2	RESTORE[4]				A,X	NZVC

1 Memory operand must reference a "String Object" with a capacity value stored in the "before byte descriptor".

2 The referenced "String Object" is modified. Also, A is corrupted such that it contains the count of the number of characters read but truncated from the object's result due to unavailable capacity. Thus, a zero value in A indicates that the entire input has been stored in the string.

3 SAVEr and RESTOREr are complementary matched operations.

4 SAVE and RESTORE are complementary matched operations.