Part I: Purpose

X-Makina is a 16-bit load-and-store RISC emulator. Along with that, there is an assembler handles the assembly code and converts them into machine code. As it is upgraded from XM2 to XM3, there is a new exception handling subsystem with several new instructions.

To support legacy software, it is essential to convert previous XM2 instructions into the new format. The goal for this project is to design a preprocessor that could translate some existing XM2 instructions into XM3 equivalent instructions to allow legacy XM2 software running well in the XM3 module.

Part II: Description of the Algorithms

MAIN

CREATE XM-3 output file

OPEN XM-2 input file

CREATE a Branch Array *Stores the label of branches*

Convert XM-2 instructions to XM-3 instructions

READ first record from XM-2 input file

WHILE (Not reach the end of file) DO

    WHILE (Not reach the end of line) DO

        GET tokens through parsing record by blank space OR tab

        NAME as 1st token, 2nd token (if exists), 3rd token (IF exists), 4th token (IF exists)

    END WHILE

    IF 1st token MATCH Legacy Bit Inst OR Legacy Branch Inst OR Legacy Other Inst list

        CALL T**RANSLATE** and PASS 1st & 2nd (IF exists) & 3rd (IF exists) token as Inst & Oper & Comm token

    ELSE IF 1st token is comment OR end of line OR Directive

        WRITE current record into XM-3 output file

    ELSE IF 2nd token exists AND MATCH Legacy Bit Inst OR Legacy Branch Inst OR Legacy Other Inst list *Case: 1st token is label*

        CALL **TRANSLATE** and PASS 1st & 2nd & 3rd (IF exists) & 4th (IF exists) token as Lbl & Inst & Oper & Comm token

    ELSE

        WRITE current record into XM-3 output file

    ENDIF

    READ next record from XM-2 input file

END WHILE

CLOSE XM-2 input file

Replace BRA with specific branch

READ first record from XM-3 file

WHILE (Not reach the end of file) DO

    WHILE (Not reach the end of line) DO

        GET tokens through parsing record by blank space OR tab

        NAME as 1st token, 2nd token (if exists), 3rd token (IF exists), 4th token (IF exists)

    END WHILE

    IF 1st token matches any items in Branch Array

        GO THROUGH till the end of file SERACH for “BRA” + 1st token

        REPLACE that record with the content of this label

    READ next record from XM-3 file

END WHILE

CLOSE XM-3 output file

TRANSLATE

IF Inst token MATCH Legacy Pri Inst list AND Oper token doesn’t exist

    FROM Legacy Pri Inst FIND corresponding token in New Pri Inst

    CREATE new record as the form:(Lbl token) + token in New Pri Inst + (Comm token)

IF Inst token MATCH Legacy Bit Inst list AND Oper token exists

    FROM Legacy Bit Inst FIND corresponding token in New Bit Inst

    CREATE new record as the form:(Lbl token) + token in New Bit Inst + (Comm token)

ELSE IF Inst token MATCH Legacy Branch Inst list

FROM Legacy Branch Inst FIND corresponding token in New Branch Inst

CREATE new record as the form:

"CEX" + token in New Branch Inst,#1,#0

"BRA" + Oper token + (Comm token)

STORE Oper token in Branch Array

ELSE IF Inst token MATCH Legacy Other Inst list*

IF Oper token does not exist

IF Inst token same as "RET"

CREATE new record as the form: (Lbl token) + "MOV" + R5,R7 + (Comm token)

ELSE

ISSUE a Warning

BREAK

ELSE IF Inst token same as "CALL"

CREATE new record as the form: (Lbl token) + "BL" + Oper token + (Comm token)

ELSE IF Inst token same as "PULL"

CREATE new record as the form: (Lbl token) + "LD" + R6+,Oper token + (Comm token)

ELSE IF Inst token same as "PUSH"

CREATE new record as the form: (Lbl token) + "ST" + Oper token,-R6 + (Comm token)

ELSE IF Inst token same as "JUMP"

CREATE new record as the form: (Lbl token) + "MOV" + Oper token,R7 + (Comm token)

ELSE IF Inst token same as "CLR.B"

CREATE new record as the form: (Lbl token) + “MOVL” + #0,Oper token + (Comm token)

ELSE IF Inst token same as “CLR” OR “CLR.W”

CREATE new record as the form: (Lbl token) + “MOVLZ” + #0,Oper token + (Comm token)

ELSE

ISSUE a Warning

BREAK

ENDIF

ELSE

ISSUE a Warning

BREAK

ENDIF

WRITE new record into XM-3 output file

EXIT

Part III: Major Data Structure

***General Concept: ***

File = (Record) *Contains all the Record*

Record = (Label) + ([Instruction | Directive]) + (Operand) + (; Comment)

Label = Alphabetic + 0 {Alphanumeric} 30

Instruction = * An instruction mnemonic *

Directive = [ALIGN | BSS | BYTE | END | EQU | ORG | WORD]

Operand = Value + 0{“,” + Operand}3

Value = [Numeric | Label]

Comment = * Start with “;” Text associated with the record – ignored by the assembler*

Token = [Label | Instruction | Directive | Operand |; Comment]

Warning = *An indicator shows the argument is missing in a record*

Register = [R0 | R1 | R2 | R3 | R4 | R5 | R6 | R7]

Rn+ = *Pop from stack*

-Rn = *Push from stack*

Array = *Container object that holds a fixed number of values of a single type*

***Instruction**: *

XM3 Instruction Set= [BL | BRA | CEX | SETPRI | SVC | SETCC | CLRCC | ADD | ADDC | SUB | SUBC | DADD | CMP | XOR | AND | BIT | BIC | BIS | MOV | SWAP | SRA | RRC | SWPB | SXT | LD | ST | SVC | MOVL | MOVLZ | MOVLS | MOVLH | LDR | STR]

Legacy Pri Inst = [SPL0 | SPL1 | SPL2 | SPL3 | SPL4 | SPL5 | SPL6 | SPL7]

New Pri Inst=[SETPRI #0 | SETPRI #1 | SETPRI #2 | SETPRI #3 | SETPRI #4 | SETPRI #5| SETPRI #6 | SETPRI #7]

Legacy Bit Inst = [SEV | SEN | SEZ | CLC | CLV | CLN | CLZ]

New Branch Inst = [CLRCC C | CLRCC N | CLRCC Z | CLRCC V | SETCC C | SETCC N | SETCC Z | SETCC V]

Legacy Branch Inst = [BEQ | BZ | BNE | BNZ | BGE | BC | BNC | BN]

New Branch Inst = [EQ | NE | GE | LT | CS | CC | MI]

Legacy Other Inst = [CALL | PULL | PUSH | RET | JUMP | CLR.B | CLR | CLR.W]

New Other Inst = [BL | LD | ST | MOV | MOVL | MOVLZ]

XM2 instruction	XM3 instruction
SPLn	SETPRI #n
CLC	CLRCC C
CLN	CLRCC N
CLZ	CLRCC Z
CLV	CLRCC V
SEC	SETCC C
SEN	SETCC N
SEZ	SETCC Z
SEV	SETCC V
BEQ|BZ label	CEX EQ,#1,#0 BRA label
BNE|BNZ label	CEX NE,#1,#0 BRA label
BGE label	CEX GE,#1,#0 BRA label
BLT label	CEX LT,#1,#0 BRA label
BC label	CEX CS,#1,#0 BRA label
BNC label	CEX CC,#1,#0 BRA label
BN label	CEX MI,#1,#0 BRA label

XM2 – XM3 Instruction Converting Table

XM2 instruction	XM3 instruction
CALL subr	BL subr
PULL arg	LD R6+,arg
PUSH arg	ST arg,-R6
RET	MOV R5,R7
JUMP arg	MOV arg,R7
CLR.B arg	MOVL #0,arg
CLR arg	MOVLZ #0,arg
CLR.W arg	MOVLZ #0,arg

***Data Type**: *

Numeric = [“$” + [Unsigned | Signed] | “‘” + Char | “#” + Hex]

Unsigned = [0 .. 65535]

Signed = [-32768 .. +0 .. +65535]

Char = [Alphanumeric | Escaped] + “‘”

Hex = 1{0 .. 9 | A .. F | a .. f} * Hex values range from #0 to #FFFF *

Escaped = “\” + Alphanumeric

Alphabetic = [A..Z | a..z | _]

Alphanumeric = [A..Z | a..z | 0..9 | _]