Instruction Decomposition

NOTION:

MAR : Memory Address Register

MBR : Memory Buffer Register

IR: Instruction Register

GPR: General Purpose Register

XR: Index Register indicator

IND: Indirect Indicator

OP1: Operand Register 1 in ALU

OP2: Operand Register 2 in ALU

RES: Result Register

M (addr): Fetch value form memory address

R (Register): Fetch value form register

1. LDR R, X, Address [,I]

//Fetch the instruction from memory

PC ==> MAR

M(MAR) ==> MBR

MBR ==> IR

//Decode the instruction

IR0-5 -- OP code

IR6-7 -- XR

IR8-9 -- GPR

IR10 -- IND

IR11 -- Trace

IR12-19 -- Addr

//Compute effective address

if IND == 0 & XR == 0;

Addr ==> MAR

if IND == 0 & XR == 1,2,3;

Addr ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

if IND == 1 & XR == 0;

Addr ==> MAR

M(MAR) ==> MBR

MBR ==> MAR

if IND == 1 & XR == 1,2,3

Addr ==> MAR

M(MAR) ==> MBR

MBR ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

//Load Register From Memory

M(MAR) ==> MBR

MBR ==> R(GPR)

// PC update

PC++ ==> PC

1. STR R, X, Address [,I]

//Fetch the instruction form memory

PC ==> MAR

M(MAR) ==> MBR

MBR ==> IR

//Decode the instruction

IR0-5 -- OP code

IR6-7 -- XR

IR8-9 -- GPR

IR10 -- IND

IR11 -- Trace

IR12-19 -- Addr

//Compute effective address

if IND == 0 & XR == 0;

Addr ==> MAR

if IND == 0 & XR == 1,2,3;

Addr ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

if IND == 1 & XR == 0;

Addr ==> MAR

M(MAR) ==> MBR

MBR ==> MAR

if IND == 1 & XR == 1,2,3

Addr ==> MAR

M(MAR) ==> MBR

MBR ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

// Store register to memory

R(GPR) ==> M(MAR)

// PC update

PC++ ==> PC

1. LDA R, X, Address, [,I]

//Fetch the instruction from memory

PC ==> MAR

M(MAR) ==> MBR

MBR ==> IR

//Decode the instruction

IR0-5 -- OP code

IR6-7 -- XR

IR8-9 -- GPR

IR10 -- IND

IR11 -- Trace

IR12-19 -- Addr

//Compute effective address

if IND == 0 & XR == 0;

Addr ==> MAR

if IND == 0 & XR == 1,2,3;

Addr ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

if IND == 1 & XR == 0;

Addr ==> MAR

M(MAR) ==> MBR

MBR ==> MAR

if IND == 1 & XR == 1,2,3

Addr ==> MAR

M(MAR) ==> MBR

MBR ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

// Load Register with Address

M(MAR) ==> R(GPR)

// PC update

PC++ ==> PC

1. AMR R, X, Address, [,I]

//Fetch the instruction from memory

PC ==> MAR

M(MAR) ==> MBR

MBR ==> IR

//Decode the instruction

IR0-5 -- OP code

IR6-7 -- XR

IR8-9 -- GPR

IR10 -- IND

IR11 -- Trace

IR12-19 -- Addr

//Compute effective address

if IND == 0 & XR == 0;

Addr ==> MAR

if IND == 0 & XR == 1,2,3;

Addr ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

if IND ==1 & XR == 0;

Addr ==> MAR

M(MAR) ==> MBR

MBR ==> MAR

if IND ==1 & XR == 1,2,3

Addr ==> MAR

M(MAR) ==> MDR

MDR ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

// Fetch the operand from the EA and compute

M(MAR) == MBR

MBR ==> OP1

R(GPR) ==> OP2

OP2 - OP1 ==> RES

RES ==> R (GPR)

// PC update

PC++ ==> PC

1. SMR R, X, Address, [,I]

//Fetch the instruction from memory

PC ==> MAR

M(MAR) ==> MBR

MBR ==> IR

//Decode the instruction

IR0-5 -- OP code

IR6-7 -- XR

IR8-9 -- GPR

IR10 -- IND

IR11 -- Trace

IR12-19 -- Addr

//Compute effective address

if IND == 0 & XR == 0;

Addr ==> MAR

if IND == 0 & XR == 1,2,3;

Addr ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

if IND ==1 & XR == 0;

Addr ==> MAR

M(MAR) ==> MBR

MBR ==> MAR

if IND ==1 & XR == 1,2,3

Addr ==> MAR

M(MAR) ==> MDR

MDR ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

// Fetch the operand from the EA and compute

M(MAR) == MBR

MBR ==> OP1

R(GPR) ==> OP2

OP2 - OP1 ==> RES

RES ==> R (GPR)

// PC update

PC++ ==> PC

1. AIR R, X, Address, [,I]

//Fetch the instruction from memory

PC ==> MAR

M(MAR) ==> MBR

MBR ==> IR

//Decode the instruction

IR0-5 -- OP code

IR6-7 -- XR

IR8-9 -- GPR

IR10 -- IND

IR11 -- Trace

IR12-19 – Immediate

//Compute effective address

if Immediate = 0

does nothing;

else

RF (RFI1) ==> OP1;

Immediate ==> OP2;

OP1+OP2 ==> OP1;

OP1 ==> RF (RFI1);

// PC update

PC++ ==> PC

1. SIR R, X, Address, [,I]

//Fetch the instruction from memory

PC ==> MAR

M(MAR) ==> MBR

MBR ==> IR

//Decode the instruction

IR0-5 -- OP code

IR6-7 -- XR

IR8-9 -- GPR

IR10 -- IND

IR11 -- Trace

IR12-19 – Immediate

//Compute effective address

If Immediate = 0

does nothing;

else

RF (RFI1) ==> OP1;

Immediate ==> OP2;

OP1-OP2 ==> OP1;

OP1 ==> RF (RFI1);

// PC update

PC++ ==> PC

1. LDX R, X, Address, [,I]

//Fetch the instruction from memory

PC ==> MAR

M(MAR) ==> MBR

MBR ==> IR

//Decode the instruction

IR0-5 -- OP code

IR6-7 -- GPR

IR8-9 -- XR

IR10 -- IND

IR11 -- Trace

IR12-19 -- ADDR

//Compute effective address

if IND == 0 & XR == 0;

Addr ==> MAR

if IND == 0 & XR == 1,2,3;

Addr ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

if IND ==1 & XR == 0;

Addr ==> MAR

M(MAR) ==> MBR

MBR ==> MAR

if IND ==1 & XR == 1,2,3

Addr ==> MAR

M(MAR) ==> MDR

MDR ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

// Load Index Register from Memory

M(MAR)==>MBR

MBR==>R(XR)

// PC update

PC++ ==> PC

1. STX R, X, Address, [,I]

//Fetch the instruction from memory

PC ==> MAR

M(MAR) ==> MBR

MBR ==> IR

//Decode the instruction

IR0-5 -- OP code

IR6-7 -- GPR

IR8-9 -- XR

IR10 -- IND

IR11 -- Trace

IR12-19 -- ADDR

//Compute effective address

if IND == 0 & XR == 0;

Addr ==> MAR

if IND == 0 & XR == 1,2,3;

Addr ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

if IND ==1 & XR == 0;

Addr ==> MAR

M(MAR) ==> MBR

MBR ==> MAR

if IND ==1 & XR == 1,2,3

Addr ==> MAR

M(MAR) ==> MDR

MDR ==> OP1

R(XR) ==> OP2

OP1+OP2 ==> RES

RES ==> MAR

// Store index register to memory

R(XR)==>M(MAR)

// PC update

PC++ ==> PC