

컨트롤 시그널 목록

Instruction Decoder

Signal	Size	Description
icode	6	instruction opcode
aluop	4	type of ALU operation
imode	1	Indication for immediate mode
zmode	1	Indication for zero-extension mode
lgmode	1	Indication for long mode
kmode	1	Indication for keep mode
rmode	1	Indication for PC-relative mode
lkmode	1	Indication for link mode
setcc	1	Set condition code reg
signed	1	indication for signed operation
unary	1	Indication for unary operation
jump	1	indication for jump instruction
branch	1	indication for branch instruction
load	1	indication for load instruction
movhi	1	indication for move high instruction
mulsel	2	indication for multiplication mode (00=mul, 01=mulh, 10=mulfx)
rD	4	reg ID of destination
rA	4	reg ID of sourceA
rB	4	reg ID of sourceB
immB	31	immediate operand
cond	4	branch condition
dmen	1	enable for data memory
dmrw	1	R/W for data memory (0 = read, 1 = write)
dmsize	2	size of data memory access (00=word, 01=byte, 10=halfword)
dmsext	1	sign extension mode for load operation
wben	1	enable for register write-back

kmode, movhi 시그널은 사용되지 않음.

ALUOP

OP	Description
ADD	0000
SUB	0001
MUL	0010
DIV	0011
MOD	0100
SHL	0101
SHR	0110
ROL	0111
ROR	1000
AND	1001
OR	1010
XOR	1011
NOT	1100

Stage Computation Table

	MOV rD, rB		MOVI{Z} rD, rB
Fetch	instr <- imem[pc] pc <= pc + 4	Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 000000 aluop <- ADD imode <- 0 zmode <- 0 lgmode <- x kmode <- x rmode <- x lkmode <- x setcc <- x signed <- x unary <- 1 jump <- 0 branch <- 0 load <- 0 movhi <- 0 mulsel <- x rD <- instr[23:20] rA <- x rB <- instr[15:12] immB <- x cond <- x dmen <- 0 dmrw <- 0 dmsize <- x dmsext <- x wben <- 1	Decode	icode <- 000000 aluop <- ADD imode <- 1 zmode <- instr[24] lgmode <- x kmode <- x rmode <- x lkmode <- x setcc <- x signed <- x unary <- 1 jump <- 0 branch <- 0 load <- 0 movhi <- 0 mulsel <- x rD <- instr[23:20] rA <- x rB <- x immB <- instr[19:0] cond <- x dmen <- 0 dmrw <- 0 dmsize <- x dmsext <- x wben <- 1
Regfile	valA <- 0 valB <- R[rB]	Regfile	valA <- 0 valB <- if(zmode) ZeroExt{immB} else SignExt{immB}
Execute	(aluCC, valE) <- valA 'aluop' valB if(setcc) NZCV <= aluCC	Execute	(aluCC, valE) <- valA 'aluop' valB if(setcc) NZCV <= aluCC
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	ADD{S} rD, rA, rB		ADDI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 000001</p> <p>aluop <- ADD</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 000001</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	SUB{S} rD, rA, rB		SUBI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 000010</p> <p>aluop <- SUB</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 000010</p> <p>aluop <- SUB</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	MOVH rD, rB		MOVHL rD, rB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 000011</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- 0</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 1</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- x</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 000011</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- 1</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 1</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- x</p> <p>rB <- x</p> <p>immB <- instr[19:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- 0</p> <p>valB <- SignExt{immB << 16}</p>	Regfile	<p>valA <- 0</p> <p>valB <- SignExt{immB << 12}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	CMP{S} rA, rB		CMPI{S} rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 000100</p> <p>aluop <- SUB</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- x</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 0</p>	Decode	<p>icode <- 000100</p> <p>aluop <- SUB</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- x</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 0</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	-	Write-Back	-

	MUL{S} rD, rA, rB		MUL{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 000101</p> <p>aluop <- MUL</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- 00</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 000101</p> <p>aluop <- MUL</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- 00</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p>(valA 'aluop' valB)[31:0]</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p>(valA 'aluop' valB)[31:0]</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	MULH{S} rD, rA, rB		MULHI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 000110</p> <p>aluop <- MUL</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 1</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- 01</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 000110</p> <p>aluop <- MUL</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 1</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- 01</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB}</p>
Execute	<p>(aluCC, valE) <-</p> <p>(valA 'aluop' valB)[63:32]</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(aluCC, valE) <-</p> <p>(valA 'aluop' valB)[63:32]</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	MULHU{S} rD, rA, rB		MULHUI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 000111</p> <p>aluop <- MUL</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 0</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- 01</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 000111</p> <p>aluop <- MUL</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 0</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- 01</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- ZeroExt{immB}</p>
Execute	<p>(aluCC, valE) <-</p> <p>(valA 'aluop' valB)[63:32]</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(aluCC, valE) <-</p> <p>(valA 'aluop' valB)[63:32]</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	MULFX{S} rD, rA, rB		MULFXI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 001000</p> <p>aluop <- MUL</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 1</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- 10</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 001000</p> <p>aluop <- MUL</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 1</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- 10</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB << 8}</p>
Execute	<p>(aluCC, valE) <-</p> <p>(valA 'aluop' valB)[47:16]</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(aluCC, valE) <-</p> <p>(valA 'aluop' valB)[47:16]</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	DIV{S} rD, rA, rB		DIVI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 001001</p> <p>aluop <- DIV</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 1</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 001001</p> <p>aluop <- DIV</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 1</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	DIVU{S} rD, rA, rB		DIVUI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 001010</p> <p>aluop <- DIV</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 0</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 001010</p> <p>aluop <- DIV</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 0</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- ZeroExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	MOD{S} rD, rA, rB		MOD{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 001011</p> <p>aluop <- MOD</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 1</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 001011</p> <p>aluop <- MOD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 1</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB}</p>
Execute	<p>(aluCC, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(aluCC, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	MODU{S} rD, rA, rB		MODUI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 001100</p> <p>aluop <- MOD</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 0</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 001100</p> <p>aluop <- MOD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 0</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- ZeroExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	SHL{S} rD, rA, rB		SHLI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 011000</p> <p>aluop <- SHL</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 011000</p> <p>aluop <- SHL</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- ZeroExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	ASR{S} rD, rA, rB		ASRI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 011001</p> <p>aluop <- SHR</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 1</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 011001</p> <p>aluop <- SHR</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 1</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- ZeroExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	LSR{S} rD, rA, rB		LSRI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 011010</p> <p>aluop <- SHR</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 0</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 011010</p> <p>aluop <- SHR</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- 0</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- ZeroExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	ROL{S} rD, rA, rB		ROL{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 011011</p> <p>aluop <- ROL</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 011011</p> <p>aluop <- ROL</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- ZeroExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	ROR{S} rD, rA, rB		RORI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 011100</p> <p>aluop <- ROR</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 011100</p> <p>aluop <- ROR</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- ZeroExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	AND{S} rD, rA, rB		ANDI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 100001</p> <p>aluop <- AND</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 100001</p> <p>aluop <- AND</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- ZeroExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	OR{S} rD, rA, rB		OR{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 100010</p> <p>aluop <- OR</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 100010</p> <p>aluop <- OR</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- ZeroExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	XOR{S} rD, rA, rB		XORI{S} rD, rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 100011</p> <p>aluop <- XOR</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 100011</p> <p>aluop <- XOR</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- ZeroExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	NOT{S} rD, rB		NOT!{S} rD, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 100100</p> <p>aluop <- NOT</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 1</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- x</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 100100</p> <p>aluop <- NOT</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 1</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- x</p> <p>rB <- x</p> <p>immB <- instr[19:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- 0</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- 0</p> <p>valB <- ZeroExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	if(wben) R[rD] <= valE	Write-Back	if(wben) R[rD] <= valE

	BCHK{S} rA, rB		BCHKI{S} rA, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 100101</p> <p>aluop <- AND</p> <p>imode <- 0</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- x</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB <- x</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 0</p>	Decode	<p>icode <- 100101</p> <p>aluop <- AND</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- x</p> <p>lkmode <- x</p> <p>setcc <- instr[24]</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- x</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 0</p> <p>dmrw <- 0</p> <p>dmsize <- x</p> <p>dmsext <- x</p> <p>wben <- 0</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- R[rB]</p>	Regfile	<p>valA <- R[rA]</p> <p>valB <- ZeroExt{immB}</p>
Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>	Execute	<p>(alucc, valE) <-</p> <p> valA 'aluop' valB</p> <p>if(setcc) NZCV <= alucc</p>
Memory	-	Memory	-
Write-Back	-	Write-Back	-

	LDR rD, rA, immB		LDRI rD, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 101000</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- 0</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 1</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 1</p> <p>dmrw <- 0</p> <p>dmsize <- 00</p> <p>dmsext <- x</p> <p>wben <- 1</p>	Decode	<p>icode <- 101000</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- 0</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 1</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 1</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- x</p> <p>rB <- x</p> <p>immB <- instr[19:0]</p> <p>cond <- x</p> <p>dmen <- 1</p> <p>dmrw <- 0</p> <p>dmsize <- 00</p> <p>dmsext <- x</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB}</p>	Regfile	<p>valA <- 0</p> <p>valB <- ZeroExt{immB}</p>
Execute	(alucc, valE) <- valA 'aluop' valB	Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]	Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM	Write-Back	if(wben) R[rD] <= valM

	LDRR rD, immB
Fetch	<pre> instr <- imem[pc] pc <= pc + 4 </pre>
Decode	<pre> icode <- 101000 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- 1 lkmode <- x setcc <- x signed <- x unary <- 0 jump <- 0 branch <- 0 load <- 1 movhi <- 0 mulsel <- x rD <- instr[23:20] rA <- x rB <- x immB <- instr[19:0] cond <- x dmen <- 1 dmrw <- 0 dmsize <- 00 dmsext <- x wben <- 1 </pre>
Regfile	<pre> valA <- pc valB <- SignExt{immB} </pre>
Execute	<pre> (alucc, valE) <- valA 'aluop' valB </pre>
Memory	<pre> if(dmen & !dmrw) valM <- M[valE] </pre>
Write-Back	<pre> if(wben) R[rD] <= valM </pre>

	LDRB rD, rA, immB		LDRBI rD, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 101001</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- 0</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 1</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 1</p> <p>dmrw <- 0</p> <p>dmsize <- 01</p> <p>dmsext <- 0</p> <p>wben <- 1</p>	Decode	<p>icode <- 101001</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- 0</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 1</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 1</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- x</p> <p>rB <- x</p> <p>immB <- instr[19:0]</p> <p>cond <- x</p> <p>dmen <- 1</p> <p>dmrw <- 0</p> <p>dmsize <- 01</p> <p>dmsext <- 0</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB}</p>	Regfile	<p>valA <- 0</p> <p>valB <- ZeroExt{immB}</p>
Execute	(alucc, valE) <- valA 'aluop' valB	Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]	Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM	Write-Back	if(wben) R[rD] <= valM

	LDRBR rD, immB
Fetch	<pre> instr <- imem[pc] pc <= pc + 4 </pre>
Decode	<pre> icode <- 101001 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- 1 lkmode <- x setcc <- x signed <- x unary <- 0 jump <- 0 branch <- 0 load <- 1 movhi <- 0 mulsel <- x rD <- instr[23:20] rA <- x rB <- x immB <- instr[19:0] cond <- x dmen <- 1 dmrw <- 0 dmsize <- 01 dmsext <- 0 wben <- 1 </pre>
Regfile	<pre> valA <- pc valB <- SignExt{immB} </pre>
Execute	<pre> (alucc, valE) <- valA 'aluop' valB </pre>
Memory	<pre> if(dmen & !dmrw) valM <- M[valE] </pre>
Write-Back	<pre> if(wben) R[rD] <= valM </pre>

	LDRSB rD, rA, immB		LDRSBI rD, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 101010</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- 0</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 1</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 1</p> <p>dmrw <- 0</p> <p>dmsize <- 01</p> <p>dmsext <- 1</p> <p>wben <- 1</p>	Decode	<p>icode <- 101010</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- 0</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 1</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 1</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- x</p> <p>rB <- x</p> <p>immB <- instr[19:0]</p> <p>cond <- x</p> <p>dmen <- 1</p> <p>dmrw <- 0</p> <p>dmsize <- 01</p> <p>dmsext <- 1</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB}</p>	Regfile	<p>valA <- 0</p> <p>valB <- ZeroExt{immB}</p>
Execute	(alucc, valE) <- valA 'aluop' valB	Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]	Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM	Write-Back	if(wben) R[rD] <= valM

	LDRSB rD, immB
Fetch	<pre> instr <- imem[pc] pc <= pc + 4 </pre>
Decode	<pre> icode <- 101010 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- 1 lkmode <- x setcc <- x signed <- x unary <- 0 jump <- 0 branch <- 0 load <- 1 movhi <- 0 mulsel <- x rD <- instr[23:20] rA <- x rB <- x immB <- instr[19:0] cond <- x dmen <- 1 dmrw <- 0 dmsize <- 01 dmsext <- 1 wben <- 1 </pre>
Regfile	<pre> valA <- pc valB <- SignExt{immB} </pre>
Execute	<pre> (alucc, valE) <- valA 'aluop' valB </pre>
Memory	<pre> if(dmen & !dmrw) valM <- M[valE] </pre>
Write-Back	<pre> if(wben) R[rD] <= valM </pre>

	LDRH rD, rA, immB		LDRHI rD, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 101011</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- 0</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 1</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 1</p> <p>dmrw <- 0</p> <p>dmsize <- 10</p> <p>dmsext <- 0</p> <p>wben <- 1</p>	Decode	<p>icode <- 101011</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- 0</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 1</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 1</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- x</p> <p>rB <- x</p> <p>immB <- instr[19:0]</p> <p>cond <- x</p> <p>dmen <- 1</p> <p>dmrw <- 0</p> <p>dmsize <- 10</p> <p>dmsext <- 0</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB}</p>	Regfile	<p>valA <- 0</p> <p>valB <- ZeroExt{immB}</p>
Execute	(alucc, valE) <- valA 'aluop' valB	Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]	Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM	Write-Back	if(wben) R[rD] <= valM

	LDRHR rD, immB
Fetch	<pre> instr <- imem[pc] pc <= pc + 4 </pre>
Decode	<pre> icode <- 101011 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- 1 lkmode <- x setcc <- x signed <- x unary <- 0 jump <- 0 branch <- 0 load <- 1 movhi <- 0 mulsel <- x rD <- instr[23:20] rA <- x rB <- x immB <- instr[19:0] cond <- x dmen <- 1 dmrw <- 0 dmsize <- 10 dmsext <- 0 wben <- 1 </pre>
Regfile	<pre> valA <- pc valB <- SignExt{immB} </pre>
Execute	<pre> (alucc, valE) <- valA 'aluop' valB </pre>
Memory	<pre> if(dmen & !dmrw) valM <- M[valE] </pre>
Write-Back	<pre> if(wben) R[rD] <= valM </pre>

	LDRSH rD, rA, immB		LDRSHI rD, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 101100</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- 0</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 1</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- instr[19:16]</p> <p>rB <- x</p> <p>immB <- instr[15:0]</p> <p>cond <- x</p> <p>dmen <- 1</p> <p>dmrw <- 0</p> <p>dmsize <- 10</p> <p>dmsext <- 1</p> <p>wben <- 1</p>	Decode	<p>icode <- 101100</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- 0</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 1</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 1</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- instr[23:20]</p> <p>rA <- x</p> <p>rB <- x</p> <p>immB <- instr[19:0]</p> <p>cond <- x</p> <p>dmen <- 1</p> <p>dmrw <- 0</p> <p>dmsize <- 10</p> <p>dmsext <- 1</p> <p>wben <- 1</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB}</p>	Regfile	<p>valA <- 0</p> <p>valB <- ZeroExt{immB}</p>
Execute	(alucc, valE) <- valA 'aluop' valB	Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]	Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM	Write-Back	if(wben) R[rD] <= valM

	LDRSHR rD, immB
Fetch	<pre> instr <- imem[pc] pc <= pc + 4 </pre>
Decode	<pre> icode <- 101100 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- 1 lkmode <- x setcc <- x signed <- x unary <- 0 jump <- 0 branch <- 0 load <- 1 movhi <- 0 mulsel <- x rD <- instr[23:20] rA <- x rB <- x immB <- instr[19:0] cond <- x dmen <- 1 dmrw <- 0 dmsize <- 10 dmsext <- 1 wben <- 1 </pre>
Regfile	<pre> valA <- pc valB <- SignExt{immB} </pre>
Execute	<pre> (alucc, valE) <- valA 'aluop' valB </pre>
Memory	<pre> if(dmen & !dmrw) valM <- M[valE] </pre>
Write-Back	<pre> if(wben) R[rD] <= valM </pre>

	STR rB, rA, immB		STRI rB, immB
Fetch	$\text{instr} \leftarrow \text{imem}[\text{pc}]$ $\text{pc} \leq \text{pc} + 4$	Fetch	$\text{instr} \leftarrow \text{imem}[\text{pc}]$ $\text{pc} \leq \text{pc} + 4$
Decode	$\text{icode} \leftarrow 101101$ $\text{aluop} \leftarrow \text{ADD}$ $\text{imode} \leftarrow 1$ $\text{zmode} \leftarrow x$ $\text{lgmode} \leftarrow x$ $\text{kmode} \leftarrow x$ $\text{rmode} \leftarrow 0$ $\text{lkmode} \leftarrow x$ $\text{setcc} \leftarrow x$ $\text{signed} \leftarrow x$ $\text{unary} \leftarrow 0$ $\text{jump} \leftarrow 0$ $\text{branch} \leftarrow 0$ $\text{load} \leftarrow 0$ $\text{movhi} \leftarrow 0$ $\text{mulsel} \leftarrow x$ $\text{rD} \leftarrow x$ $\text{rA} \leftarrow \text{instr}[19:16]$ $\text{rB} \leftarrow \text{instr}[15:12]$ $\text{immB} \leftarrow \{\text{instr}[23:20], \text{instr}[11:0]\}$ $\text{cond} \leftarrow x$ $\text{dmen} \leftarrow 1$ $\text{dmrw} \leftarrow 1$ $\text{dmsize} \leftarrow 00$ $\text{dmsext} \leftarrow x$ $\text{wben} \leftarrow 0$	Decode	$\text{icode} \leftarrow 101101$ $\text{aluop} \leftarrow \text{ADD}$ $\text{imode} \leftarrow 1$ $\text{zmode} \leftarrow x$ $\text{lgmode} \leftarrow x$ $\text{kmode} \leftarrow x$ $\text{rmode} \leftarrow 0$ $\text{lkmode} \leftarrow x$ $\text{setcc} \leftarrow x$ $\text{signed} \leftarrow x$ $\text{unary} \leftarrow 1$ $\text{jump} \leftarrow 0$ $\text{branch} \leftarrow 0$ $\text{load} \leftarrow 0$ $\text{movhi} \leftarrow 0$ $\text{mulsel} \leftarrow x$ $\text{rD} \leftarrow x$ $\text{rA} \leftarrow x$ $\text{rB} \leftarrow \text{instr}[15:12]$ $\text{immB} \leftarrow \{\text{instr}[23:16], \text{instr}[11:0]\}$ $\text{cond} \leftarrow x$ $\text{dmen} \leftarrow 1$ $\text{dmrw} \leftarrow 1$ $\text{dmsize} \leftarrow 00$ $\text{dmsext} \leftarrow x$ $\text{wben} \leftarrow 0$
Regfile	$\text{valA} \leftarrow \text{R}[\text{rA}]$ $\text{valB} \leftarrow \text{SignExt}\{\text{immB}\}$	Regfile	$\text{valA} \leftarrow 0$ $\text{valB} \leftarrow \text{ZeroExt}\{\text{immB}\}$
Execute	$(\text{alucc}, \text{valE}) \leftarrow$ $\quad \text{valA} \text{ 'aluop' } \text{valB}$	Execute	$(\text{alucc}, \text{valE}) \leftarrow$ $\quad \text{valA} \text{ 'aluop' } \text{valB}$
Memory	$\text{if}(\text{dmen} \& \text{dmrw})$ $\quad \text{M}[\text{valE}] \leq \text{R}[\text{rB}]$	Memory	$\text{if}(\text{dmen} \& \text{dmrw})$ $\quad \text{M}[\text{valE}] \leq \text{R}[\text{rB}]$
Write-Back	-	Write-Back	-

	STRB rB, immB
Fetch	<pre> instr <- imem[pc] pc <= pc + 4 </pre>
Decode	<pre> icode <- 101101 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- 1 lkmode <- x setcc <- x signed <- x unary <- 0 jump <- 0 branch <- 0 load <- 0 movhi <- 0 mulsel <- x rD <- x rA <- x rB <- instr[15:12] immB <- {instr[23:16], instr[11:0]} cond <- x dmen <- 1 dmrw <- 1 dmsize <- 00 dmsext <- x wben <- 0 </pre>
Regfile	<pre> valA <- pc valB <- SignExt{immB} </pre>
Execute	<pre> (alucc, valE) <- valA 'aluop' valB </pre>
Memory	<pre> if(dmen & dmrw) M[valE] <= R[rB] </pre>
Write-Back	-

	STRB rB, rA, immB		STRB rB, immB
Fetch	instr <- imem[pc] pc <= pc + 4	Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101110 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- 0 lkmode <- x setcc <- x signed <- x unary <- 0 jump <- 0 branch <- 0 load <- 0 movhi <- 0 mulsel <- x rD <- x rA <- instr[19:16] rB <- instr[15:12] immB <- {instr[23:20], instr[11:0]} cond <- x dmen <- 1 dmrw <- 1 dmsize <- 01 dmsext <- x wben <- 0	Decode	icode <- 101110 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- 0 lkmode <- x setcc <- x signed <- x unary <- 1 jump <- 0 branch <- 0 load <- 0 movhi <- 0 mulsel <- x rD <- x rA <- x rB <- instr[15:12] immB <- {instr[23:16], instr[11:0]} cond <- x dmen <- 1 dmrw <- 1 dmsize <- 01 dmsext <- x wben <- 0
Regfile	valA <- R[rA] valB <- SignExt{immB}	Regfile	valA <- 0 valB <- ZeroExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB	Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & dmrw) M[valE] <= R[rB]	Memory	if(dmen & dmrw) M[valE] <= R[rB]
Write-Back	-	Write-Back	-

	STRBR rB, immB
Fetch	<pre> instr <- imem[pc] pc <= pc + 4 </pre>
Decode	<pre> icode <- 101110 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- 1 lkmode <- x setcc <- x signed <- x unary <- 0 jump <- 0 branch <- 0 load <- 0 movhi <- 0 mulsel <- x rD <- x rA <- x rB <- instr[15:12] immB<- {instr[23:16],instr[11:0]} cond <- x dmen <- 1 dmrw <- 1 dmsize <- 01 dmsext <- x wben <- 0 </pre>
Regfile	<pre> valA <- pc valB <- SignExt{immB} </pre>
Execute	<pre> (alucc, valE) <- valA 'aluop' valB </pre>
Memory	<pre> if(dmen & dmrw) M[valE] <= R[rB] </pre>
Write-Back	-

	STRH rB, rA, immB		STRHI rB, immB
Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>	Fetch	<p>instr <- imem[pc]</p> <p>pc <= pc + 4</p>
Decode	<p>icode <- 101111</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- 0</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 0</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- x</p> <p>rA <- instr[19:16]</p> <p>rB <- instr[15:12]</p> <p>immB<- {instr[23:20],instr[11:0]}</p> <p>cond <- x</p> <p>dmen <- 1</p> <p>dmrw <- 1</p> <p>dmsize <- 10</p> <p>dmsext <- x</p> <p>wben <- 0</p>	Decode	<p>icode <- 101111</p> <p>aluop <- ADD</p> <p>imode <- 1</p> <p>zmode <- x</p> <p>lgmode <- x</p> <p>kmode <- x</p> <p>rmode <- 0</p> <p>lkmode <- x</p> <p>setcc <- x</p> <p>signed <- x</p> <p>unary <- 1</p> <p>jump <- 0</p> <p>branch <- 0</p> <p>load <- 0</p> <p>movhi <- 0</p> <p>mulsel <- x</p> <p>rD <- x</p> <p>rA <- x</p> <p>rB <- instr[15:12]</p> <p>immB<- {instr[23:16],instr[11:0]}</p> <p>cond <- x</p> <p>dmen <- 1</p> <p>dmrw <- 1</p> <p>dmsize <- 10</p> <p>dmsext <- x</p> <p>wben <- 0</p>
Regfile	<p>valA <- R[rA]</p> <p>valB <- SignExt{immB}</p>	Regfile	<p>valA <- 0</p> <p>valB <- ZeroExt{immB}</p>
Execute	(alucc, valE) <- valA 'aluop' valB	Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & dmrw) M[valE] <= R[rB]	Memory	if(dmen & dmrw) M[valE] <= R[rB]
Write-Back	-	Write-Back	-

	STRHR rB, immB
Fetch	<pre> instr <- imem[pc] pc <= pc + 4 </pre>
Decode	<pre> icode <- 101111 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- 1 lkmode <- x setcc <- x signed <- x unary <- 0 jump <- 0 branch <- 0 load <- 0 movhi <- 0 mulsel <- x rD <- x rA <- x rB <- instr[15:12] immB<- {instr[23:16],instr[11:0]} cond <- x dmen <- 1 dmrw <- 1 dmsize <- 10 dmsext <- x wben <- 0 </pre>
Regfile	<pre> valA <- pc valB <- SignExt{immB} </pre>
Execute	<pre> (alucc, valE) <- valA 'aluop' valB </pre>
Memory	<pre> if(dmen & dmrw) M[valE] <= R[rB] </pre>
Write-Back	-

	B{cond} immB
Fetch	<pre> instr <- imem[pc] pclnc <- pc + 4 pc <= taken ? valE : pclnc </pre>
Decode	<pre> icode <- 110010 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- 1 lkmode <- x setcc <- x signed <- x unary <- 0 jump <- 0 branch <- 1 load <- 0 movhi <- 0 mulsel <- x rD <- x rA <- x rB <- x immB <- instr[25:4] cond <- instr[3:0] dmen <- 0 dmrw <- 0 dmsize <- x dmsext <- x wben <- 0 </pre>
Regfile	<pre> valA <- pc valB <- SignExt{immB} </pre>
Execute	<pre> valE <- valA 'aluop' valB Taken <- eval (NZCV, cond) </pre>
Memory	-
Write-Back	-

	JMP rA, immB		JMPI rA, immB
Fetch	$\text{instr} \leftarrow \text{imem}[pc]$ $\text{pclnc} \leftarrow pc + 4$ $pc \leq valE$	Fetch	$\text{instr} \leftarrow \text{imem}[pc]$ $\text{pclnc} \leftarrow pc + 4$ $pc \leq valE$
Decode	$icode \leftarrow 110011$ $aluop \leftarrow ADD$ $imode \leftarrow 1$ $zmode \leftarrow x$ $lgmode \leftarrow x$ $kmode \leftarrow x$ $rmode \leftarrow x$ $lkmode \leftarrow 0$ $setcc \leftarrow x$ $signed \leftarrow x$ $unary \leftarrow 0$ $jump \leftarrow 1$ $branch \leftarrow 0$ $load \leftarrow 0$ $movhi \leftarrow 0$ $mulsel \leftarrow x$ $rD \leftarrow x$ $rA \leftarrow \text{instr}[19:16]$ $rB \leftarrow x$ $immB \leftarrow \text{instr}[15:0]$ $cond \leftarrow x$ $dmen \leftarrow 0$ $dmrw \leftarrow 0$ $dmsize \leftarrow x$ $dmsext \leftarrow x$ $wben \leftarrow 0$	Decode	$icode \leftarrow 110011$ $aluop \leftarrow ADD$ $imode \leftarrow 1$ $zmode \leftarrow x$ $lgmode \leftarrow x$ $kmode \leftarrow x$ $rmode \leftarrow x$ $lkmode \leftarrow 0$ $setcc \leftarrow x$ $signed \leftarrow x$ $unary \leftarrow 1$ $jump \leftarrow 1$ $branch \leftarrow 0$ $load \leftarrow 0$ $movhi \leftarrow 0$ $mulsel \leftarrow x$ $rD \leftarrow x$ $rA \leftarrow x$ $rB \leftarrow x$ $immB \leftarrow \text{instr}[23:0]$ $cond \leftarrow x$ $dmen \leftarrow 0$ $dmrw \leftarrow 0$ $dmsize \leftarrow x$ $dmsext \leftarrow x$ $wben \leftarrow 0$
Regfile	$valA \leftarrow R[rA]$ $valB \leftarrow \text{ZeroExt}\{immB\}$	Regfile	$valA \leftarrow 0$ $valB \leftarrow \text{ZeroExt}\{immB\}$
Execute	$valE \leftarrow valA \text{ 'aluop' } valB$	Execute	$valE \leftarrow valA \text{ 'aluop' } valB$
Memory	-	Memory	-
Write-Back	-	Write-Back	-

	JMPL rA, immB		JMPIL rA, immB
Fetch	$\text{instr} \leftarrow \text{imem}[pc]$ $\text{pclnc} \leftarrow pc + 4$ $pc \leq valE$	Fetch	$\text{instr} \leftarrow \text{imem}[pc]$ $\text{pclnc} \leftarrow pc + 4$ $pc \leq valE$
Decode	$icode \leftarrow 110011$ $aluop \leftarrow ADD$ $imode \leftarrow 1$ $zmode \leftarrow x$ $lgmode \leftarrow x$ $kmode \leftarrow x$ $rmode \leftarrow x$ $lkmode \leftarrow 1$ $setcc \leftarrow x$ $signed \leftarrow x$ $unary \leftarrow 0$ $jump \leftarrow 1$ $branch \leftarrow 0$ $load \leftarrow 0$ $movhi \leftarrow 0$ $mulsel \leftarrow x$ $rD \leftarrow \text{instr}[23:20]$ $rA \leftarrow \text{instr}[19:16]$ $rB \leftarrow x$ $immB \leftarrow \text{instr}[15:0]$ $cond \leftarrow x$ $dmen \leftarrow 0$ $dmrw \leftarrow 0$ $dmsize \leftarrow x$ $dmsext \leftarrow x$ $wben \leftarrow 1$	Decode	$icode \leftarrow 110011$ $aluop \leftarrow ADD$ $imode \leftarrow 1$ $zmode \leftarrow x$ $lgmode \leftarrow x$ $kmode \leftarrow x$ $rmode \leftarrow x$ $lkmode \leftarrow 1$ $setcc \leftarrow x$ $signed \leftarrow x$ $unary \leftarrow 1$ $jump \leftarrow 1$ $branch \leftarrow 0$ $load \leftarrow 0$ $movhi \leftarrow 0$ $mulsel \leftarrow x$ $rD \leftarrow \text{instr}[23:20]$ $rA \leftarrow x$ $rB \leftarrow x$ $immB \leftarrow \text{instr}[19:0]$ $cond \leftarrow x$ $dmen \leftarrow 0$ $dmrw \leftarrow 0$ $dmsize \leftarrow x$ $dmsext \leftarrow x$ $wben \leftarrow 1$
Regfile	$valA \leftarrow R[rA]$ $valB \leftarrow \text{ZeroExt}\{immB\}$	Regfile	$valA \leftarrow 0$ $valB \leftarrow \text{ZeroExt}\{immB\}$
Execute	$valE \leftarrow valA \text{ 'aluop' } valB$	Execute	$valE \leftarrow valA \text{ 'aluop' } valB$
Memory	-	Memory	-
Write-Back	$R[rD] \leq pclnc$	Write-Back	$R[rD] \leq pclnc$