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
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
Regfile	valA <- 0 valB <- R[rB]
Execute	(alucc, valE) <- valA 'aluop' valB if(setcc) NZCV <= alucc
Memory	-
Write-Back	if(wben) R[rD] <= valE

	MOVI{Z} rD, rB
Fetch	instr <- imem[pc] pc <= pc + 4
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 <- if(zmode) ZeroExt{immB} else SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB if(setcc) NZCV <= alucc
Memory	-
Write-Back	if(wben) R[rD] <= valE

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

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

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

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

	MOVH rD, rB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 000011 aluop <- ADD imode <- 1 zmode <- x lgmode <- 0 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[15:0] cond <- x dmen <- 0 dmrw <- 0 dmsize <- x dmsext <- x wben <- 1
Regfile	valA <- 0 valB <- SignExt{immB << 16}
Execute	(alucc, valE) <- valA 'aluop' valB if(setcc) NZCV <= alucc
Memory	-
Write-Back	if(wben) R[rD] <= valE

	MOVHL rD, rB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 000011 aluop <- ADD imode <- 1 zmode <- x lgmode <- 1 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 <- SignExt{immB << 12}
Execute	(alucc, valE) <- valA 'aluop' valB if(setcc) NZCV <= alucc
Memory	-
Write-Back	if(wben) R[rD] <= valE

	CMP{S} rA, rB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 000100 aluop <- SUB imode <- 0 zmode <- x lgmode <- x kmode <- x rmode <- x lkmode <- x setcc <- instr[24] 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 <- x cond <- x dmen <- 0 dmrw <- 0 dmsize <- x dmsext <- x wben <- 0
Regfile	valA <- R[rA] valB <- R[rB]
Execute	(alucc, valE) <- valA 'aluop' valB if(setcc) NZCV <= alucc
Memory	-
Write-Back	-

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

	NOT{S} rD, rB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 100100 aluop <- NOT imode <- 0 zmode <- x lgmode <- x kmode <- x rmode <- x lkmode <- x setcc <- instr[24] 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
Regfile	valA <- 0 valB <- R[rB]
Execute	(alucc, valE) <- valA 'aluop' valB if(setcc) NZCV <= alucc
Memory	-
Write-Back	if(wben) R[rD] <= valE

	NOTI{S} rD, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 100100 aluop <- NOT imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- x lkmode <- x setcc <- instr[24] 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 <- ZeroExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB if(setcc) NZCV <= alucc
Memory	-
Write-Back	if(wben) R[rD] <= valE

	BCHK{S} rA, rB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 100101 aluop <- AND imode <- 0 zmode <- x lgmode <- x kmode <- x rmode <- x lkmode <- x setcc <- instr[24] 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 <- x cond <- x dmen <- 0 dmrw <- 0 dmsize <- x dmsext <- x wben <- 0
Regfile	valA <- R[rA] valB <- R[rB]
Execute	(alucc, valE) <- valA 'aluop' valB if(setcc) NZCV <= alucc
Memory	-
Write-Back	-

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

	LDR rD, rA, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101000 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 <- 1 movhi <- 0 mulsel <- x rD <- instr[23:20] rA <- instr[19:16] rB <- x immB <- instr[15:0] cond <- x dmen <- 1 dmrw <- 0 dmsize <- 00 dmsext <- x wben <- 1
Regfile	valA <- R[rA] valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRI rD, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101000 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 <- 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
Regfile	valA <- 0 valB <- ZeroExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRR rD, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	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
Regfile	valA <- pc valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRB rD, rA, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101001 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 <- 1 movhi <- 0 mulsel <- x rD <- instr[23:20] rA <- instr[19:16] rB <- x immB <- instr[15:0] cond <- x dmen <- 1 dmrw <- 0 dmsize <- 01 dmsext <- 0 wben <- 1
Regfile	valA <- R[rA] valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRBI rD, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101001 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 <- 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
Regfile	valA <- 0 valB <- ZeroExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRBR rD, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	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
Regfile	valA <- pc valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRSB rD, rA, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101010 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 <- 1 movhi <- 0 mulsel <- x rD <- instr[23:20] rA <- instr[19:16] rB <- x immB <- instr[15:0] cond <- x dmen <- 1 dmrw <- 0 dmsize <- 01 dmsext <- 1 wben <- 1
Regfile	valA <- R[rA] valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRSBI rD, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101010 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 <- 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
Regfile	valA <- 0 valB <- ZeroExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRSBR rD, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	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
Regfile	valA <- pc valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRH rD, rA, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101011 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 <- 1 movhi <- 0 mulsel <- x rD <- instr[23:20] rA <- instr[19:16] rB <- x immB <- instr[15:0] cond <- x dmen <- 1 dmrw <- 0 dmsize <- 10 dmsext <- 0 wben <- 1
Regfile	valA <- R[rA] valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRHI rD, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101011 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 <- 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
Regfile	valA <- 0 valB <- ZeroExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRHR rD, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	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
Regfile	valA <- pc valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRSH rD, rA, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101100 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 <- 1 movhi <- 0 mulsel <- x rD <- instr[23:20] rA <- instr[19:16] rB <- x immB <- instr[15:0] cond <- x dmen <- 1 dmrw <- 0 dmsize <- 10 dmsext <- 1 wben <- 1
Regfile	valA <- R[rA] valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRSHI rD, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101100 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 <- 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
Regfile	valA <- 0 valB <- ZeroExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	LDRSHR rD, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	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
Regfile	valA <- pc valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & !dmrw) valM <- M[valE]
Write-Back	if(wben) R[rD] <= valM

	STR rB, rA, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101101 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 <- 00 dmsext <- x wben <- 0
Regfile	valA <- R[rA] valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & dmrw) M[valE] <= R[rB]
Write-Back	-

	STRI rB, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101101 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 <- 00 dmsext <- x wben <- 0
Regfile	valA <- 0 valB <- ZeroExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & dmrw) M[valE] <= R[rB]
Write-Back	-

	STRR rB, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	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
Regfile	valA <- pc valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & dmrw) M[valE] <= R[rB]
Write-Back	-

	STRB rB, rA, immB
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
Regfile	valA <- R[rA] valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & dmrw) M[valE] <= R[rB]
Write-Back	-

	STRBI rB, immB
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 <- 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 <- 0 valB <- ZeroExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & dmrw) M[valE] <= R[rB]
Write-Back	-

	STRBR rB, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	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
Regfile	valA <- pc valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & dmrw) M[valE] <= R[rB]
Write-Back	-

	STRH rB, rA, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101111 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 <- 10 dmsext <- x wben <- 0
Regfile	valA <- R[rA] valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & dmrw) M[valE] <= R[rB]
Write-Back	-

	STRHI rB, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	icode <- 101111 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 <- 10 dmsext <- x wben <- 0
Regfile	valA <- 0 valB <- ZeroExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & dmrw) M[valE] <= R[rB]
Write-Back	-

	STRHR rB, immB
Fetch	instr <- imem[pc] pc <= pc + 4
Decode	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
Regfile	valA <- pc valB <- SignExt{immB}
Execute	(alucc, valE) <- valA 'aluop' valB
Memory	if(dmen & dmrw) M[valE] <= R[rB]
Write-Back	-

	B{cond} immB
Fetch	instr <- imem[pc] pcInc <- pc + 4 pc <= taken ? valE : pcInc
Decode	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
Regfile	valA <- pc valB <- SignExt{immB}
Execute	valE <- valA 'aluop' valB Taken <- eval (NZCV, cond)
Memory	-
Write-Back	-

	JMP rB
Fetch	instr <- imem[pc] pcInc <- pc + 4 pc <= valB
Decode	icode <- 110011 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- x lkmode <- 0 setcc <- x signed <- x unary <- 1 jump <- 1 branch <- 0 load <- 0 movhi <- 0 mulsel <- x rD <- x rA <- x rB <- instr[15:12] immB<- x cond <- x dmen <- 0 dmrw <- 0 dmsize <- x dmsext <- x wben <- 0
Regfile	valA <- 0 valB <- R[rB]
Execute	-
Memory	-
Write-Back	-

	JMPI immB
Fetch	instr <- imem[pc] pcInc <- pc + 4 pc <= valB
Decode	icode <- 110011 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- x lkmode <- 0 setcc <- x signed <- x unary <- 1 jump <- 1 branch <- 0 load <- 0 movhi <- 0 mulsel <- x rD <- x rA <- x rB <- x immB<- instr[23:0] cond <- x dmen <- 0 dmrw <- 0 dmsize <- x dmsext <- x wben <- 0
Regfile	valA <- 0 valB <- ZeroExt{immB}
Execute	-
Memory	-
Write-Back	-

	JMPL rD, rB
Fetch	instr <- imem[pc] pcInc <- pc + 4 pc <= valB
Decode	icode <- 110011 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- x lkmode <- 1 setcc <- x signed <- x unary <- 1 jump <- 1 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
Regfile	valA <- x valB <- R[rB]
Execute	-
Memory	-
Write-Back	R[rD] <= pcInc

	JMPIL rD, immB
Fetch	instr <- imem[pc] pcInc <- pc + 4 pc <= valB
Decode	icode <- 110011 aluop <- ADD imode <- 1 zmode <- x lgmode <- x kmode <- x rmode <- x lkmode <- 1 setcc <- x signed <- x unary <- 1 jump <- 1 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 <- ZeroExt{immB}
Execute	-
Memory	-
Write-Back	R[rD] <= pcInc