ARM tiles for FMJ

Note 1: The tiles marked with ++ are those you must use in your instruction selection program (HW12), and all other tiles are optional for HW12. This is to simplify the homework and this simplified set should be "enough" to select instructions for FMJ programs. Note that the use of this set is not going to produce the best (optimized) instructions. So if you feel adventurous, please go ahead using more tiles.

Note 2: Const(imm8_k) is an imm8 constant which is an 8-bit number with a shift. We assume all constant integers appear in an FMJ program are imm8 numbers. (If your instruction selection program sees a constant that is not imm8, then an error is encountered.) Const(2^k) is a special imm8 number (i.e., a one-bit with shift) and your maxMunch program should try to recognize it in order to decrease the number of instructions generated. Note that imm8 may be a negative number if the left-most bit (of the 32 bits) is 1 (for example, 0xff000000 represents .

Note 3: Const(Offset) is a number between –4095 to +4095, used in ldr and str instructions.

Note 3: Tiles with * are not necessary if all arithmetic operations that involve constants only and whose result is also imm8 numbers are alreay eliminated in the IR tree. For example, we assume Binop(*, 100, 2) is already replaced with Const(200) in the IR tree. Otherwise, those tiles with * are still useful.

	add destReg, srcReg, op2
1++	Binop(+,e1,e2)
2	Binop(+,e1,Const(imm8_k))
3	Binop(+,Const(imm8_k),e1)
4	Binop(+,e1,Binop(*,e2,Const(2^k)))
5*	Binop(+,e1,Binop(*,Const(imm8_k), Const(2^k)))
6	Binop(+,e1,Binop(*,Const(2^k),e2))
7*	Binop(+,e1,Binop(*,Const(2^k), Const(imm8_k)))
8	Binop(+,Binop(*,e1,Const(2^k)),e2)
9*	Binop(+,Binop(*,Const(imm8_k),Const(2^k)), e2)
10	Binop(+,Binop(*,Const(2^k),e1),e2)
11*	Binop(+,Binop(*,Const(2^k),Const(imm8_k)),e2)
12	Binop(+,e1,Binop(/,e2,Const(2^k)))
13*	Binop(+,e1,Binop(/,Const(imm8_k), Const(2^k)))
14	Binop(+,Binop(/,e1,Const(2^k)), e2)
15*	Binop(+,Binop(/,Const(imm8_k), Const(2^k)),e2)

	sub destReg, srcReg, op2
1++	Binop(-,e1,e2)
2	Binop(-,e1,Const(imm8_k))
3	Binop(-,e1,Binop(*, e2, Const(2^k)))
4*	Binop(-,e1,Binop(*, Const(imm8_k), Const(2^k)))
5	Binop(-,e1,Binop(*, Const(2^k), e2))
6*	Binop(-,e1,Binop(*, Const(2^k), Const(imm8_k)))
7	Binop(-,e1,Binop(/, e2, Const(2^k)))
8*	Binop(-,e1,Binop(/, Const(imm8_k), Const(2^k)))

	rsb destReg, srcReg, op2
1	Binop(-,e1,e2)
2	Binop(-,Const(imm8_k),e1)
3	Binop(-,Binop(*,e1,Const(2^k)),e2)
4*	Binop(-,Binop(*, Const(imm8_k),Const(2^k)),e2)
5	Binop(-,Binop(*,Const(2^k),e1),e2)
6*	Binop(-,Binop(*,Const(2^k),Const(imm8_k)),e2)
7	Binop(-,Binop(/,e1,Const(2^k)),e2)
8*	Binop(-,Binop(/,Const(imm8_k),Const(2^k)),e2)

	mul destReg, srcReg1, srcReg2
1++	Binop(*,e1,e2)

	and/orr destReg, srcReg, op2
null	These are for bit operations. Not used for FMJ.

	mov destReg, op2
1++	Move(Temp,e1)
2++	Move(Temp,Binop(* or /,e1,Const(2^k)))
3++	Binop(* or /, e1, Const(2^k))
4	Move(Temp,Binop(*,Const(2^k),e1))
5	Binop(*, Const(2^k), e1)
6	Move(Temp, Const(imm8_k))
7++	Const(imm8_k)
8	Move(Temp,Binop(* or /, Const(imm8_k), Const(2^k)))
9	Binop(* or /, Const(imm8_k), Const(2^k)))
10	Move(Temp, Binop(*, Const(2^k), Const(imm8_k)))
11	Binop(*,Const(2^k),Const(imm8_k))

	ldr destReg, [locationReg[, offset]]
1++	Mem(e1)
2++	Mem(Binop(+,e1, Const(Offset))
3++	Mem(Binop(-,e1, Const(Offset)))
4	Mem(Binop(+,Const(Offset),e1))
5	Mem(Binop(+,e1,e2))
6	Move(Temp,Mem(e1))
7	Move(Temp,Mem(Binop(+,e1,Const(Offset))))
8	Move(Temp,Mem(Binop(-,e1,Const(Offset))))
9	Move(Temp,Mem(Binop(+,Const(Offset),e1)))

	ldr destReg, =<label>
1++	Move(Temp,Name)

	str srcReg, [locationReg[, offset]]
1++	Move(Mem(e1),e2)
2++	Move(Mem(Binop(+,e1,Const(Offset))),e2)
3++	Move(Mem(Binop(-,e1,Const(Offset))),e2)
4	Move(Mem(Binop(+,Const(Offset),e1)),e2)

	cmp Reg1, op2
null	no tile for this alone. See special case below.

	label: nop
1++	Label(Temp_label)

	b label
1++	Jump(Name)

	bx reg
1++	Jump(e1)

	bl label + return-label: nop
1++	Seq(Move(lr, Name(return-label), Jump(Name), Label(return-label))

	blx reg + ft-label: nop
1++	Seq(Move(lr, Name(return-label)), Jump(e), Label(return-label))

Note: Below co stands for gt, ge, lt, le, e, and ne, and co' for the inverse of co (e.g., if co=gt, then co'=le).

The following tiles should be the regular case in which CJump is followed by the false label (as the result of the canonicalization process). You should try to use these tiles before using the tiles in the next table.

	cmp reg1,op2 + bco true-label + false-label: nop
1++	Cjump(co,e1,e2,true-label,false-label) + Label(false-label)
2	Cjump(co,e1,Const(imm8_k),true-label,false-label) + Label(false-label)
3	Cjump(co',Const(imm8_k),e1,false-label,true-label)+ Label(false-label)
4	Cjump(co,e1,Binop(*,e2,Const(2^k)),true-label,false-label)+ Label(false-label)
5*	Cjump(co,e1,Binop(*,Const(imm8_k),Const(2^k)),true-label,false-label)+ Label(false-label)
6	Cjump(co,e1,Binop(*,Const(2^k),e2),true-label,false-label) + Label(false-label)
7*	Cjump(co,e1,Binop(*,Const(2^k),Const(imm8_k)),true-label,false-label) + Label(false-label)
8	Cjump(co',Binop(*,Const(2^k),e1),e2,false-label, true-label)+ Label(false-label)
9*	Cjump(co',Binop(*,Const(2^k),Const(imm8_k)),e2,false-label, true-label) + Label(false-label)
10	Cjump(co', Binop(*,e1,Const(2^k)),e2,false-label,true-label)+ Label(false-label)
11*	Cjump(co', Binop(*, Const(imm8_k),Const(2^k)),e2,false-label,true-label) + Label(false-label)
12	Cjump(co,e1,Binop(/,e2,Const(2^k)),true-label,false-label) + Label(false-label)
13*	Cjump(co,e1,Binop(/,Const(imm8_k),Const(2^k)),true-label,false-label) + Label(false-label)
14	Cjump(co',Binop(/,e1,Const(2^k)),e2,false-label,true-label) + Label(false-label)
15*	Cjump(co',Binop(/,Const(imm8_k),Const(2^k)),e2,false-label,true-label) + Label(false-label)

The following tiles should be used only when the IR node following the Cjump is NOT the false label.

	cmp reg1,op2 + bco true-label + b false-label
1++	Cjump(co,e1,e2,true-label,false-label)
2	Cjump(co,e1,Const(imm8_k),true-label,false-label)
3	Cjump(co',Const(imm8_k),e1,false-label,true-label)
4	Cjump(co,e1,Binop(*,e2,Const(2^k)),true-label,false-label)
5*	Cjump(co,e1,Binop(*,Const(imm8_k),Const(2^k)),true-label,false-label)
6	Cjump(co,e1,Binop(*,Const(2^k),e2),true-label,false-label)
7*	Cjump(co,e1,Binop(*,Const(2^k),Const(imm8_k)),true-label,false-label)
8	Cjump(co',Binop(*,Const(2^k),e1),e2,false-label, true-label)
9*	Cjump(co',Binop(*,Const(2^k),Const(imm8_k)),e2,false-label, true-label)
10	Cjump(co', Binop(*,e1,Const(2^k)),e2,false-label,true-label)
11*	Cjump(co', Binop(*, Const(imm8_k),Const(2^k)),e2,false-label,true-label)
12	Cjump(co,e1,Binop(/,e2,Const(2^k)),true-label,false-label)
13*	Cjump(co,e1,Binop(/,Const(imm8_k),Const(2^k)),true-label,false-label)
14	Cjump(co',Binop(/,e1,Const(2^k)),e2,false-label,true-label)
15*	Cjump(co',Binop(/,Const(imm8_k),Const(2^k)),e2,false-label,true-label)