	Test Cases						
Test #	**Instruction**	**Code**	**Expected Result**	**Expected Result (Binary)**	**Test Condition**	**Summary**	**Success?**
1	MOV AL r0,#1 ADD AL r1,r0 SYS AL	4001 0110 9800	reg1 = 1	reg0 = 0000000000000001 reg1 = 0000000000000001	Register dependence	1. r0=1 2. r1=r0+r1 3. Expected result: r1==1	yes
2	MOV AL r0,#4 ADD AL r1,r0 SYS AL	4004 0110 9800	reg1 = 4	reg0 = 0000000000000100 reg1 = 0000000000000100	Register dependence	1. Set r0=4 2. r1=r1+r0 3. Expected result: r1==4	yes
3	MUL S r0,#0 MOV EQ r1,#1 ADD AL r1,r0 SYS AL	4a00 4411 0110 9800	reg1 = 1	reg1 = 0000000000000001	Conditional and register dependencies	1. r0 = r0*0; ZeroFlag=1 2. if(ZeroFlag) r1=1 3. r1=r0+r1 4. Expected result: r1==1	yes
4	MOV S r0,#0 MOV NE r1,#2 MOV AL r1,#4 MOV AL r0,r1 SYS AL	4200 4612 4014 4101 9800	reg0 = 4	reg0 = 0000000000000100 reg1 = 0000000000000100	Conditional and register dependencies	1. r0=0; ZeroFlag=1 2. if(!ZeroFlag) r1=2 3. if(ZeroFlag) r1=4 4. r0=r1 5. Expected result: r0==4	yes
5	MOV AL r0,#2 MOV AL r1,#5 MOV S r2,#0 MUL EQ r0,r1 MOV AL r2,r0 SYS AL	4002 4015 4220 4d01 4120 9800	reg2 = 10	reg2 = 0000000000001010	Conditional dependencies and multiplication	1. r0=2 2. r1=5 3. r2=0; ZeroFlag=1 4. if(ZeroFlag) r0=r0*r1 5. r2=r0 6. Expected result: r2==10	yes
6	MOV AL r0,#5 MOV AL r1,#1 STR AL r0,[r1] LDR AL r2,[r1] SYS AL	4005 4011 7901 3921 9800	reg2 = 5	reg2 = 0000000000000101	Memory dependence	1. r0=5 2. r1=1 3. memory[r1]=r0 4. r2=memory[r1] 5. Expected result: r2==5	yes
7	MOV AL r0,#6 MOV AL r1,#1 MOV S r2,#0 STR EQ r0,[r1] LDR AL r3,[r1] SYS AL	4006 4011 4220 7d01 3931 9800	reg3 = 6	reg3 = 0000000000000110	Conditional and memory dependencies	1. r0=6 2. r1=1 3. r2=0; ZeroFlag=1 4. if(ZeroFlag) memory[r1]=r0 5. r3=memory[r1] 6. Expected result: r3==6	yes
8	ADD AL r0, #6 ADD AL r1, #1 MOV AL r15, #6 SYS AL SYS AL SYS AL ADD AL r1, r0 ADD AL r0, #5 ADD AL r0, #6 ADD AL r0, #5 ADD AL r1, r0 SYS AL	0006 0011 40f6 9800 9800 9800 0110 0005 0006 0005 0110 9800	reg0 = 22 reg1 = 29	reg0 = 0000000000010110 reg1 = 0000000000011101	Jumps	1. r0=6 2. r1=r1+1 3. r15(instruction#)=7 4. End Program 5. End Program 6. End Program 7. r1=r1+r0 8. r0=r0+5 9. r0=r0+6 10. r0=r0+5 11. r1=r1+r0 12. Expected result: r0==22; r1==29	yes

9	MOV AL r3, #7 MOV AL r0, #1 SUB S r3, r0 MOV EQ r15, #6 ADD NE r1, #1 MOV AL r15, #1 MOV AL r2, #42 SYS AL	4037 4001 8b30 44f6 0611 40f1 c002 402a 9800	reg0 = 1 reg1 = 6 reg2 = 42 reg3 = 0	reg0 = 0000000000000001 reg1 = 0000000000000110 reg2 = 0000000000101010 reg3 = 0000000000000000	Jumps with loop	1. r3=7 2. r0=1 3. r3=r3-r0; ZeroFlag=0 4. if(ZeroFlag) r15(instruction#)=7 5. if(!ZeroFlag) r1=r1+1 6. r15(instruction#)=1 (jump back to line 2) 7. r2=42 (next line is carryover because constant>7) 8. Expected result: r0==1; r1==6; r2==42; r3==0	yes
10	PRE AL #1 PRE AL #2 ADD AL r0, #1 SYS AL	c001 c002 0001 9800	reg0=33	reg0 = 0000000000100001	Two PRE's in a row	1. PRE(top 12 bits of next constant)=1 2. PRE(top 12 bits of next constant)=2 3. r0=r0+1+32 (because 2 in PRE) 4. Expected result: r0==33	yes
11	ITOF AL r0,#6 ITOF AL r1,#0 ITOF AL r2,#-3 ITOF AL r3,#-7 SYS AL	3006 3010 302d 3039 9800	reg0 = 4’x40c0 reg1 = 4’x0 reg2 = 4’xc040 reg3 = 4’xc0e0	reg0 = 0100000011000000 reg1 = 0000000000000000 reg2 = 1100000001000000 reg3 = 1100000011100000	Int to float conversion	1. r0=float(6) 2. r1=float(1) 3. r2=float(-3) 4. r3=float(-7) 5. Expected result: proper conversion to bitwise float	yes
12	ITOF AL r0,#42 ITOF AL r1,#-117 SYS AL	c002 300a cff8 301b 9800	reg0 = 0x4228 reg1 = 0xc2ea	reg0 = 0100001000101000 reg1 = 1100001011101010	Int to float conversion with 'long' constants (C>7 or C<-8)	1. r0=float(42) (next line carryover because 42>7) 2. r1=float(-117) (next line carryover because -117<-8) 3. Expected result: proper conversion to bitwise float	yes
13	ITOF AL r0,#6 MOV AL r1, #0 FTOI AL r2, r0 SYS AL	3006 4010 2920 9800	reg0 = 4’x40c0 reg2 = 4’x0006	reg0 = 0100000011000000 reg2 = 0000000000000110	Float to int conversion	1. r0=float(6) 2. r1=0 3. r2=int(r0) 4. Expected result: r2==6	yes
14	ITOF AL r0,#-117 MOV AL r1, #0 FTOI AL r2, r0 SYS AL	cff8 300b 4010 2920 9800	reg0 = 4’xc2ea reg2 = 4’xff8b	reg0 = 1100001011101010 reg2 = 1111111110001011	Float to int conversion with negative constants and 'long' constants	1. r0=float(-117) (next line carryover because -117<-8) 2. r1=0 3. r2=int(r0) 4. Expected result: r2==-117	yes
15	ITOF AL r0,#5 ITOF AL r1, #6 MULF AL r0, r1 FTOI AL r2, r0 SYS AL	3005 3016 5101 2920 9800	reg0 = 16’b0100000111110000 reg1 = 16’b0100000011000000 reg2 = 16’b0000000000011110	reg0 = 0100000111110000 reg1 = 0100000011000000 reg2 = 0000000000011110	Floating point multiplication	1. r0=float(5) 2. r1=float(6) 3. r0=r0*r1 4. r2=int(r0) 5. Expected result: r2==30	yes
16	ITOF AL r0,#-10 ITOF AL r1,#7 MULF AL r0, r1 FTOI AL r2, r0 SYS AL	cfff 3006 3017 5101 2920 9800	reg0 = 16’b1100001010001100 reg1 = 16’b0100000011100000 reg2 = 16’b1111111110111010	reg0 = 1100001010001100 reg1 = 0100000011100000 reg2 = 1111111110111010	Floating point multiplication between positive and negative value	1. r0=float(-10) 2. r1=float(7) 3. r0=r0*r1 4. r2=float(r0) 5. Expected result: r2==-70	yes

17	ITOF AL r0,#-21 ITOF AL r1,#-20 MULF AL r0, r1 FTOI AL r2, r0 SYS AL	cffe 300b cffe 301c 5101 2920 9800	reg0 = 16’b0100001111010010 reg1 = 16’b1100000110100000 reg2 = 16’b0000000110100100	reg0 = 0100001111010010 reg1 = 1100000110100000 reg2 = 0000000110100100	Floating point multiplication between 'long' negative values	1. r0=-21 2. r1=-20 3. r0=r0*r1 4. r2=float(r0) 5. Expected result: r2==420	yes
18	ITOF AL r0,#5 RECF AL r1,r0 SYS AL	3005 6910 9800	reg0 = 16’b0100000010100000 reg1 = 16’b0011101101001100	reg0 = 0100000010100000 reg1 = 0011101101001100	Floating point reciprocal	1. r0=5 2. r1=reciprocal(r0) 3. Expected result: r1==0.2	no; lack of precision
19	ITOF AL r1, #-50 FTOI AL r0, r1 SYS AL	cffc 301e 2901 9800	reg0 = 16'b1111111111001110 reg1 = 16'b1100001001001000	reg0 = 1111111111001110 reg1 = 1100001001001000	Float to int conversion for a 'long' negative value	1. r1=float(-50) 2. r0=int(r1) 3. Expected result: r0==-50	yes
20	ITOF AL r0,#300 ITOF AL r1,#500 ADDF AL r0,r1 SYS AL	c012 300c c01f 3014 0901 9800	reg0 = 4'x4448	reg0 = 0100010001001000	Floating point addition on 'long' positive values	1. r0=float(300) 2. r1=float(500) 3. r0=r0+r1 4. Expected result: r0==float(800)	yes
21	ITOF AL r0,#-13 ITOF AL r1,#55 ADDF AL r0,r1 SYS AL	cfff 3003 c003 3017 0901 9800	reg0 = 4'x4228	reg0 = 0100001000101000	Floating point addition between a positive and a negative	1. r0=float(-13) 2. r1=float(55) 3. r0=r0+r1 4. Expected result: r0==float(42)	yes
22	ITOF AL r0,#-1 ITOF AL r1,#2 ITOF AL r2,#-3 ITOF AL r3,#4 ADDF AL r0,r1 ADDF AL r0,r2 ADDF AL r0,r3 FTOI AL r3,r0 SYS AL	300f 3012 302d 3034 0901 0902 0903 2930 9800	reg3 = 2	reg3 = 0000000000000010	Floating point addition	1. r0=float(-1) 2. r1=float(2) 3. r2=float(-3) 4. r3=float(4) 5. r0=r0+r1 6. r0=r0+r2 7. r0=r0+r3 8. r3=int(r0) 9. Expected result: r3==2	yes
23	ITOF AL r0,#1 ITOF AL r1,#-2 SUBF AL r0,r1 FTOI AL r3,r0 SYS AL	3001 301e 9101 2930 9800	reg3=3	reg3 = 0000000000000011	Floating point subtraction	1. r0=float(1) 2. r1=float(-2) 3. r0=r0-r1 4. r3=int(r0) 5. Expected result: r3==3	yes

24	ITOF AL r0,#-5 ITOF AL r1,#4 ITOF AL r2,#-3 ITOF AL r3,#2 ADDF AL r0,r1 MULF AL r0,r2 SUBF S r0,r3 MOV EQ r3, #17 ITOF NE r3, #0 MULF S r2, r3 FTOI EQ r3, r0 FTOI EQ r0, r0 ADD EQ r0, #41 SYS AL	300b 3014 302d 3032 0901 5102 9303 e001 4431 3630 5323 2d30 2d00 e002 0409 9800	reg0 = 42 reg2 = 0 reg3 = 1	reg0 = 0000000000101010 reg1 = 0000000000000000 reg2 = 0000000000000001	Float instructions with conditional dependencies	1. r0=float(-5) 2. r1=float(4) 3. r2=float(-3) 4. r3=float(2) 5. r0=r0+r1 6. r0=r0*r2 7. r0=r0-r3; ZeroFlag=0 8. if(ZeroFlag) r3=17 9. if(!ZeroFlag) r3=float(0) 10. r2=r2*r3; ZeroFlag=1 11. if(ZeroFlag) r3=int(r0) 12. if(ZeroFlag) r0=int(r0) 13. if(ZeroFlag) r0=r0+41 14. Expected result: r0==42; r2==0; r3==1	yes
25	MOV AL r0, #1 SLT S r0, #2 MOV EQ r3, #2 MOV NE r3, #1 SYS AL	4001 8202 4432 4631 9800	reg3=1	reg3 = 0000000000000001	Set less than when a < b	1. r0=1 2. r0 = (r0<2); ZeroFlag = (r0==0) 3. if(ZeroFlag) r3=2 4. if(!ZeroFlag) r3=1 5. Expected result: r3==1	yes
26	MOV AL r0, #3 SLT S r0, #2 MOV EQ r3, #2 MOV NE r3, #1 SYS AL	4003 8202 4432 4631 9800	reg3=2	reg3 = 0000000000000010	Set less than when a > b	1. r0=3 2. r0 = (r0<2); ZeroFlag = (r0==0) 3. if(ZeroFlag) r3=2 4. if(!ZeroFlag) r3=1 5. Expected result: r3==2	yes
27	MOV EQ r0, #42 MOV S r1, #100 MOV AL r2, #1 MOV AL r3, #2 SYS AL	e002 440a d006 4214 4021 4032 9800	reg0 = 0 reg1 = 100 reg2 = 1 reg3 = 2	reg0 = 0000000000000000 reg1 = 0000000001100100 reg2 = 0000000000000001 reg3 = 0000000000000010	Using Set in conjunction with large constants	1. if(ZeroFlag) r0=42 2. r1=100; ZeroFlag=0 3. r2=1 4. r3=2 5. Expected result: r0==0; r1==100; r2==1; r3==2	yes
28	MOV AL r0, #42 ADD S r0, #45 MOV EQ r3, #2 MOV NE r3, #1 SYS AL	c002 400a d002 020d 4432 4631 9800	reg0 = 87 reg1 = 0 reg2 = 0 reg3 = 1	reg0 = 0000000001010111 reg1 = 0000000000000000 reg2 = 0000000000000000 reg3 = 0000000000000001	Conditional codes with large constants	1. r0=42 2. r0=r0+45; ZeroFlag=0 3. if(ZeroFlag) r3=2 4. if(!ZeroFlag) r3=1 5. Expected result: r0==87; r3==1	yes

29	MOV AL r0, #42 SLT S r0, #45 MOV EQ r3, #2 MOV NE r3, #1 SYS AL	c002 400a d002 820d 4432 4631 9800	reg0 = 1 reg1 = 0 reg2 = 0 reg3 = 1	reg0 = 0000000000000001 reg1 = 0000000000000000 reg2 = 0000000000000000 reg3 = 0000000000000001	Set less than in conjunction with PRE values	1. r0=42 2. r0=(r0<45); ZeroFlag=0 3. if(ZeroFlag) r3=2 4. if(!ZeroFlag) r3=1 5. Expected result: r0=1; r3=1	yes
30	MOV AL r0, #-3 SLT AL r0, #2 SYS AL	400d 8002 9800	reg0 = 1	reg0 = 0000000000000001	Set Less Than when values are negative	1. r0=-3 2. r0=(r0<2) 3. Expected result: r0=1	yes
31	MOV AL r0, #30 MOV AL r2, r0 MOV AL r1, r0 MOV AL r0, r0 SLT S r1, #1 MOV NE r15, #11 SUB AL r0, #5 MOV AL r1, r0 MOV AL r15, #4 MOV AL r0, r0 MOV S r0, r0 MOV EQ r3, #1 MOV AL r1, r2 MOV AL r0, r0 SLT S r1, #1 MOV NE r15, #23 SUB AL r2, #3 MOV AL r1, r2 MOV AL r15, #15 MOV AL r0, r0 MOV S r2, r2 MOV EQ r2, #1 MOV NE r2, #0 SYS AL	c001 400e 4120 4110 4100 8211 f000 46fb 8805 4110 40f4 4100 4300 4431 4112 4100 8211 f001 46f7 8823 4112 c000 40ff 4100 4322 4421 4620 9800	reg2 = 1 reg3 = 1	reg0 = 0000000000000001 reg1 = 0000000000000001	FizzBuzz using 30 (reg1 is Fizz; reg2 is Buzz)	1. r0=30 2. r2=r0 3. r1=r0 4. r0=r0 //NOP, used for jumps in case instr 1 takes extra instr 5. r1=(r1<1); set ZeroFlag 6. if(!ZeroFlag) Jump to #11 7. r0=r0-5 8. r1=r0 9. Jump to #5 10. r0=r0 //NOP 11. r0=r0; set ZeroFlag //if r0==0, ZeroFlag=1 12. if(ZeroFlag) r3=1 //this is BUZZ 13. r1=r2 14. r0=r0 //NOP 15. r1=(r1<1); set ZeroFlag 16. if(!ZeroFlag) Jump to #21 17. r2=r2-3 18. r1=r2 19. Jump to #15 20. r0=r0 //NOP 21. r2=r2; setZeroFlag //if r2==0, ZeroFlag=1 22. if(ZeroFlag) r2=1 //this is FIZZ 23. if(!ZeroFlag) r2=0 24. Expected result: r2=FIZZ; r3=BUZZ	yes

32	ITOF AL r0, #6 ITOF AL r1, #2 RECF AL r1, r1 MULF AL r0, r1 FTOI AL r3, r0 SYS AL	3006 3012 6911 5101 2930 9800	reg3 = 3	reg3 = 0000000000000011	Reciprocal using one half	1. r0=float(6) 2. r1=float(2) 3. r1=1/2 4. r0=r0*r1 5. r3=int(r0) 6. Expected result: r3=3	Yes
33	ITOF AL r0, #6 ITOF AL r1, #3 RECF AL r1, r1 MULF AL r0, r1 FTOI AL r3, r0 SYS AL		reg3 = 2	reg3 = 0000000000000010	Reciprocal using one third	1. r0=float(6) 2. r1=float(3) 3. r1=1/3 4. r0=r0*r1 5. r3=int(r0) 6. Expected result: r3=2	No; lack of precision