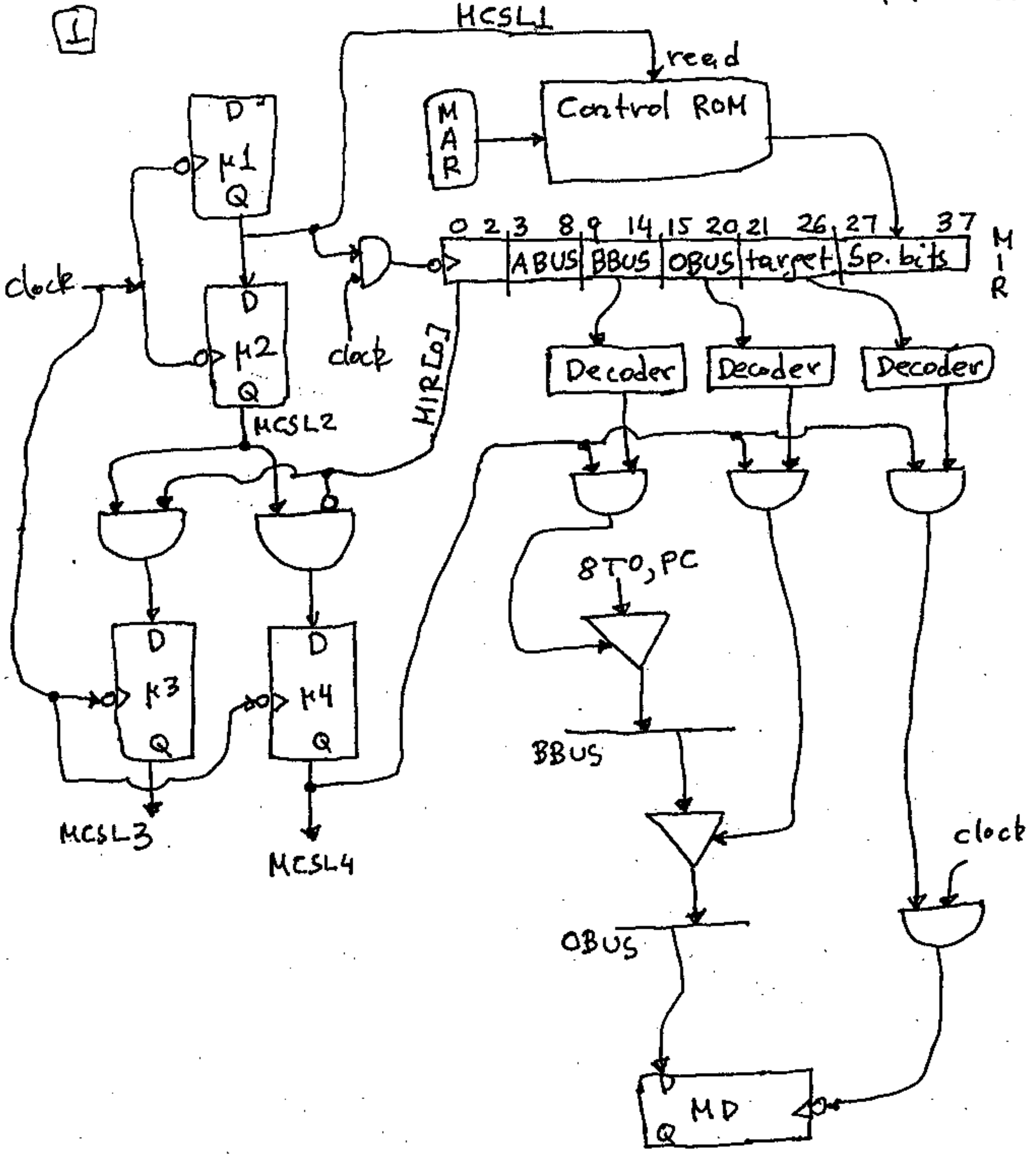
EE 3755, Spring 2003

Solutions of HW#6

[redacted]

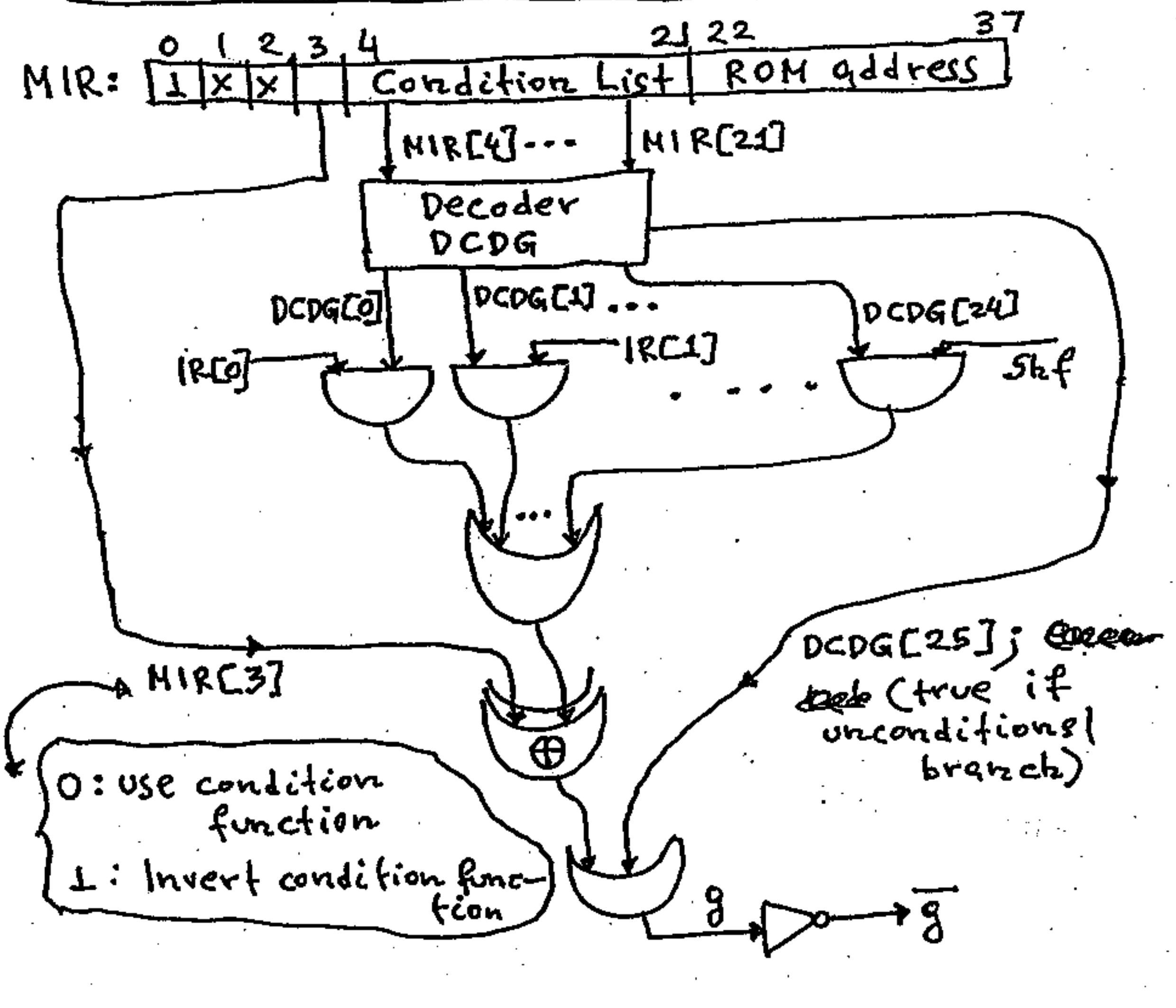
① [redacted] EE 3755, [redacted] Solutions of Hw # [redacted] 6

$\boxed{1}$



2 This was done in class. See page 21 of handout entitled "Microprogram-Based Controllers". That page is repeated below.

Implementation of the function "g" needed for the branch microinstruction


0 1 2 3 4 21 22 37
MIR: 1 x x Condition List ROM address
MIR[4] ...
MIR[21]
Decoder DCDG
DCDG[0]
DCDG[1] ...
DCDG[24]
IR[0]
IR[1]
Shf
...
MIR[3]
0: use condition function
1: Invert condition function
DCDG[25]; (true if unconditional branch)
⊕
g
g̅

3

Address in hex	Microinstruction in MICRAL
2000	→(IR[9])/(2006).
2001	$A=\overline{32T0}$; B=8T0,SP; OB=ADD; cin=0; MA ← OB.
2002	$A=\overline{32T0}$; B=8T0,SP; OB=ADD; cin=0; SP←OB.
2003	B=AC; OB=B; MD←OB.
2004	WRITE.
2005	→(1000)
2006	B=8T0,SP; OB=B; MA←OB.

2007	READ; A=32T1; B=8T0,SP; OB=ADD;
	cin=0; SP← OB.

For the above I assumed that the microcode responsible for step 100 starts at ROM address 1000 hex

The microcode in MICRAL responsible for step 77 is located in ROM addresses 2004 and 2005 hex; (look above). The microcode in binary is shown below.

ROM address in hex	Microinstruction in binary
2004	0XX XXXXXX XXXXXX XXXXXX $\underbrace{001100}_{\text{no target affected}}$ $\underbrace{00\ 000\ 0X}_{\text{flags not clocked}}$ XX 0 $\underset{\text{write}}{1}$

ROM address in hex	Microinstruction in binary
2005	$\underbrace{\mathrm{1xxx}}\ \underbrace{000000000000011001}_{\text{unconditional branching}}\ \underbrace{0001000000000000}_{\text{binary equivalent of 1000 hex}}$

$\boxed{4}$ I assume that after a jump register instruction, register transfer instruction, or Format B ADD instruction has been fetched, the controller went to control steps 100, 200, 300, respectively, to execute them. Also DCDA and DCDB are two 5-to-32 decoders. The execution of the Format B load instruction starts at step 400, while the execution of the Load lower immediate instruction starts at step 500

→ go to next page

ⓐ 100 BBUS = (R0 ! R1 ! .... ! R31) * DCDA(IR[6:10]);
OBUS = BBUS; PC ← OBUS; → (1)

ⓑ 200 BBUS = (R0 ! R1 ! ... ! R31) * DCDA(IR[6:10]);
OBUS = BBUS;
(R0 ! R1 ! ... ! R31) * DCDB(IR[11:15]) ← OBUS;
zff ← $\overline{\vee/\text{OBUS}}$; nff ← OBUS[0]; → (1).

ⓒ

300 ABUS = (16T0, IR[16:31] ! $\overline{\text{16T0}}$, IR[16:31]) * ($\overline{\text{IR[16]}}$, IR[16]);
BBUS = (R0 ! R1 ! ... ! R31) * DCDA(IR[6:10]); Cin = 0;
OBUS = ADD[1:32](ABUS; BBUS; Cin);
(R0 ! R1 ! ... ! R31) * DCDB(IR[11:15]) ← OBUS;
cff ← ADD[0](ABUS; BBUS; cin); zff ← $\overline{\vee/\text{OBUS}}$;
nff ← OBUS[0];
vff ← (ABUS[0] ∧ BBUS[0] ∧ $\overline{\text{ADD[1](ABUS; BBUS; cin)}}$)
∨ ($\overline{\text{ABUS[0]}}$ ∧ $\overline{\text{BBUS[0]}}$ ∧ ADD[1](ABUS; BBUS; cin));
→ (1).

(d)

400 ABUS=(16T0, IR[16:31] ! $\overline{16T0}$, IR[16:31]) * ($\overline{IR[16]}$, IR[16]);
BBUS=(R0 ! R1 ! ... ! R31) * DCDA(IR[6:10]);
cin = 0; OBUS= ADD[1:32](ABUS; BBUS; cin);
MA ← OBUS. " memory-address is stored in MA".

401 ADBUS=MA; read=1; MD ← DBUS. "read memory".

402 ABUS=MD; OBUS=ABUS;
(R0 ! R1 ! ... ! R31) * DCDA(IR[11:15]) ← OBUS;
zff ← $\overline{\vee/OBUS}$; nff ← OBUS[0]; → (1).
" reg. specified by F2 is loaded with memory data".

(e)

500 ABUS = IR; OBUS = ABUS;
(R0[16:31] ! R1[16:31] ! ... ! R31[16:31] * DCDA(IR[11:15])
← OBUS[16:31];
→ (1).

(5)

Instruction i : R7 ← R5 ⊕ R7
Instruction i+1 : R5 ← R7 ⊕ R5
Instruction i+2 : R7 ← R5 ⊕ R7