Writing the microprogram

Problem 1: Starting at ROM address 0100 hex, microcode in binary steps 1, 2, 3, 4 of the "First example computer" AHPL control sequence.

Solution: The steps 1, 2, 3, 4 are

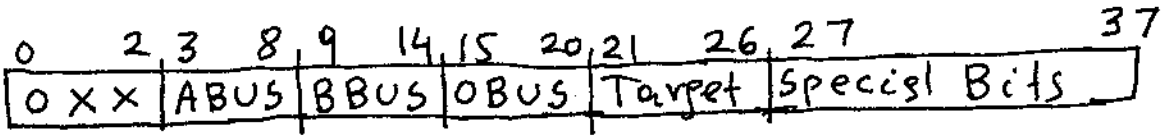
1 MA ← PC; shf ← 0
2 ADBUS = MA; read = 1; MD ← DBUS; PC ← INC(PC).
3 IR ← MD.
4 → (IR[4])/(50).

The above steps are rewritten in a way showing explicit BUS-connections

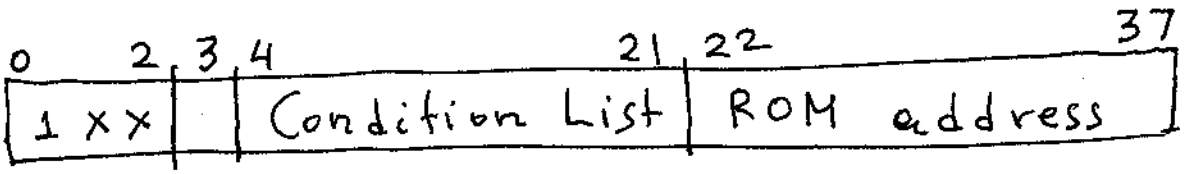
1 BBUS = 8TO, PC; OBUS = BBUS; MA ← OBUS[8:31];
shf ← 0.
2 ADBUS = MA; read = 1; MD ← DBUS;
ABUS = 32T1; BBUS = 8TO, PC; cin = 0;
OBUS = ADD[1:32](ABUS; BBUS; cin);
PC ← OBUS[8:31]
3 ABUS = MD; OBUS = ABUS; IR ← OBUS.
4 → (IR[4])/(50).

We can now translate the above into binary microcode. I will assume that microcode responsible for step 50 starts at ROM address 258A hex.

I will assume the two different types of 38-bit microinstructions namely transfer and branch the formats of which are repeated below:



(a) Transfer Microinstruction



{ 0: Use condition function
1: Invert condition function }

(b) Branch Microinstruction

I will also assume the list of BUS-connections, target selections, special bits (flag activities etc...) and branch functions offered by the handout entitled: "Microprogram-Based Controllers".

(M)

The microprogram in binary is shown below

ROM address in hex	Microinstruction in binary; (ROM contents)
0100	0XX XXXXXX 000000 000000 000000 00 000 10 XX 00
0101	0XX 000000 000000 000001 000001 00 000 0X 00 10
0102	0XX 000001 XXXXXX 000010 000010 00 000 0X XX 00
0103	1XX1 00000000000000100 0010 0101 1000 1010

ABUS BBUS oBUS target nff;zff vff cff shf cin read write

specifies IR[4]

2 5 8 A hex

ROM address

MICRAL (Microassembly Language)

Writing the microprogram in binary is difficult. One would rather microcode using a more human-oriented language called "Microassembly Language" or "MICRAL". Then, a piece of software called "Microassembler" can translate MICRAL into binary microcode.

The MICRAL looks very much like AHPL and is explained below:

- ROM addresses are expressed as 4-digit hex addresses

ROM address	Microinstruction in MICRAL
125A	A = ...; B = ...; OB = ADD; cin = 0; AC ← OB; cff ← ADD[0]; NZ; V.
137B	A = ...; B = ...; OB = AND; AC ← OB; NZ.
2123	READ "Main memory reqd"

ROM address in hex	Microinstruction in MICRAL
2536	WRITE "Main memory write"
3000	→ (1256) "Unconditional branching to microinstruction located at ROM address 1256 hex"
3125	→ ($\overline{IR[5]}$) / (1008) "Conditional branching

In the previous:

- A; B; OB denote ABUS; BBUS and OBUS respectively
- NZ denotes that nff and zff must be clocked.
- V denotes that vff must be clocked.

Problem 2: Write in MICRAL the micro-code responsible for steps 31, 32, 33, 34, 35 of the "first example computer's" AHPL control sequence.

- ROM addresses should be given in hex.
- Start your microcode at address 1000 hex

The steps 31, ..., 35 are shown below

```
31 → (IR[5])/(35)
32 MA ← DEC(SP); SP ← DEC(SP).
33 MD ← 8TO, PC.
34 ADBUS = MA; DBUS = MD; write = 1.
35 ABUS = (16TO, IR[16:31] ! 16TO, IR[16:31] ! 8TO, IR[8:31])
        *(IR[16] ∧ IR[0], IR[16] ∧ IR[0], IR[0]);
   BBUS = 8TO, PC; cin = 0;
   OBUS = (ABUS ! ADD[1:32] (ABUS; BBUS; cin)) * (IR[0], IR[0]);
   PC ← OBUS[8:31];
   → (1).
```

(a) Microcode in MICRAL

	ROM address	Microinstruction
step 31	1000	→ (~~IR[5]~~)/(1005).
step 32	1001	A = ~~32TO~~; B = 8TO, SP; OB = ADD; cin = 0; MA ← OB
	1002	A = ~~32TO~~; B = 8TO, SP; OB = ADD; cin = 0; SP ← OB
step 33	1003	B = 8TO, PC; OB = B; MD ← OB
step 34	1004	WRITE
step 35	1005	→ (~~IR[0]~~)/(1008)
	1006	A = 8TO, IR[8:31]; B = 8TO, PC; OB = A; PC ← OB.
	1007	→ (000#). "(Microcode for step 1 begins at ROM address 000# hex)"
	1008	→ (~~IR[16]~~)/(100B)
	1009	A = ~~16TO~~, IR[16:31]; B = 8TO, PC; cin = 0; OB = ADD; PC ← OB.
	100A	→ (000#).
	100B	A = 16TO, IR[16:31]; B = 8TO, PC; cin = 0; OB = ADD; PC ← OB.
	100C	→ (000#).

(b) Microinstruction located at address 1004 hex in binary

ROM address in hex	Microinstruction in binary
1004	0XX XXXXXX XXXXXX XXXXXX 001100 00 000 0X XX 0 1

no target

flags not clocked

write