جامعة بيرزيت

BIRZEIT UNIVERSITY

# ANSWER BOOKLET

Student: Digital Number 12

Course: Department: Number:

Division: Instructor:

Date: Day Month Year

**For Instructor's Use**

| Question | Grade |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| **Total** | |

State Table

| Present State | | Input | Next State | | Output |
|---|---|---|---|---|---|
| A | B | x | A | B | y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

- In general, if we have in a circuit m flip-flops, n inputs, o outputs

$\Rightarrow$

- state table has $2^{m+n}$ rows
- next state section has m columns (one for each flip-flop
- output section has o columns.

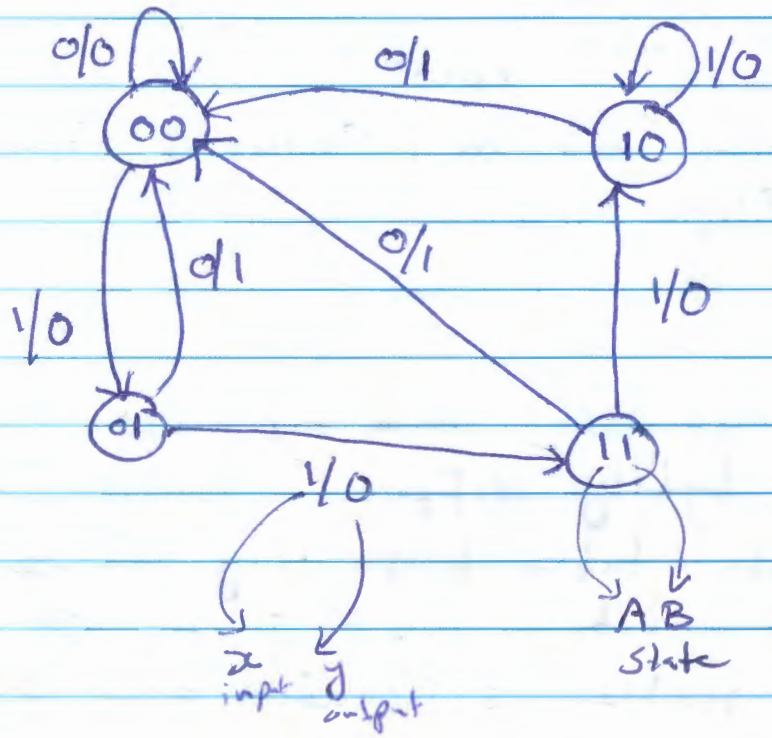
(*) sometimes the state table is presented is a slightly different form. In this case the state table has only three sections: present state, next state and output. (The input section is now included in the next state section).

| Present state | | Next state | | | | Output | |
|---|---|---|---|---|---|---|---|
| | | $x=0$ | | $x=1$ | | | |
| A | B | A | B | A | B | $x=0$ | $x=1$ |
| | | | | | | $y$ | $y$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |

## ⊛ State Diagram :

- The information available in a state table can be represented graphically in the form of a state diagram.

(This is only another way to see things)



## Flip-Flop Input Equations (Excitation Equations)

$$\left.\begin{aligned} D_A &= Ax + Bx \\ D_B &= A'x \end{aligned}\right\} \text{input equations}$$

for DFF input equations are the same as state equations because the characteristic equations $Q(t+1) = D_Q$.

Ex

Fig 5-17

Solution

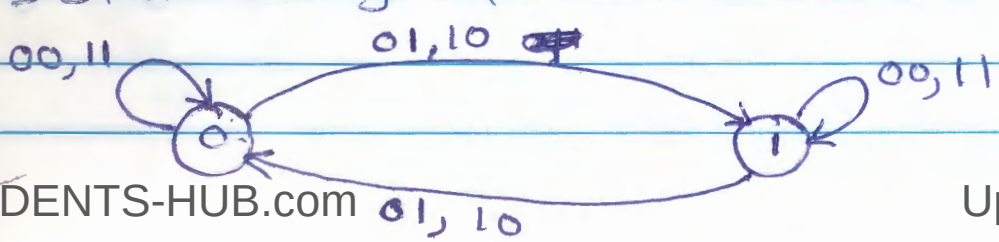
$D_A = A \oplus x \oplus y$ (input equation (excitation)

$\Rightarrow A(t+1) = A \oplus x \oplus y$ (state equation (transition)

$\Rightarrow$ The state table (Transition Table)

| Present State | Inputs | | Next State (no output) |
|---|---|---|---|
| A | x | y | A |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

$\Rightarrow$ State Diagram



## Ex: Fig 5-18

Input equations: $J_A = B$ $\quad K_A = Bx'$

$J_B = x$ $\quad K_B = A'x + Ax'$

$= A \oplus x$

State equations: by using the characteristic equation of DFF

$$Q(t+1) = JQ' + K'Q$$

$$\Rightarrow A(t+1) = BA' + (Bx')'A$$

$$= A'B + AB' + Ax$$

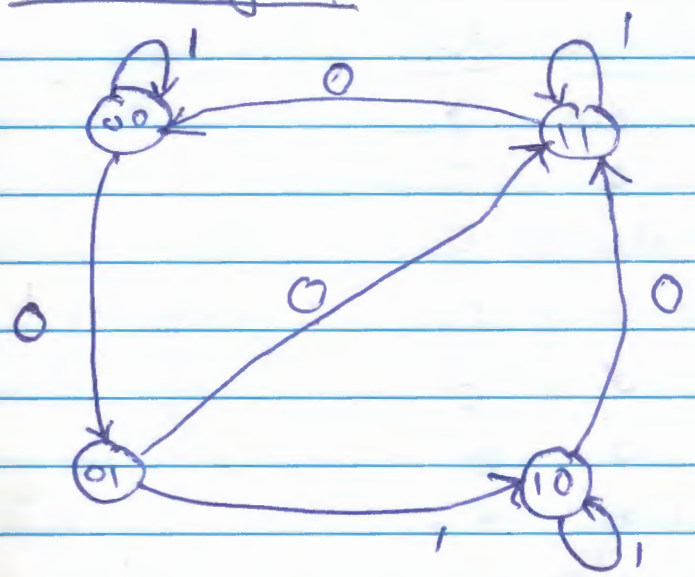
& $B(t+1) = x'B' + (A \oplus x)'B$

$$= B'x' + ABx + A'Bx'$$

State Table

| Present State | | Input | Next State | | Flip-Flop Inputs | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | FF$_A$ | | FF$_B$ | |
| A | B | x | A | B | $J_A$ | $K_A$ | $J_B$ | $K_B$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

State Diagram:




Ex. Fig 5-20

Input Equations

$T_A = Bx$ $\quad T_B = x$

Output Equation

$y = AB$ (function of present states only)

State equation: By using T Flip Flop characteristic equation $Q(t+1) = T \oplus Q = T'Q + TQ'$

$\Rightarrow A(t+1) = T_A' \cdot A + T_A A'$

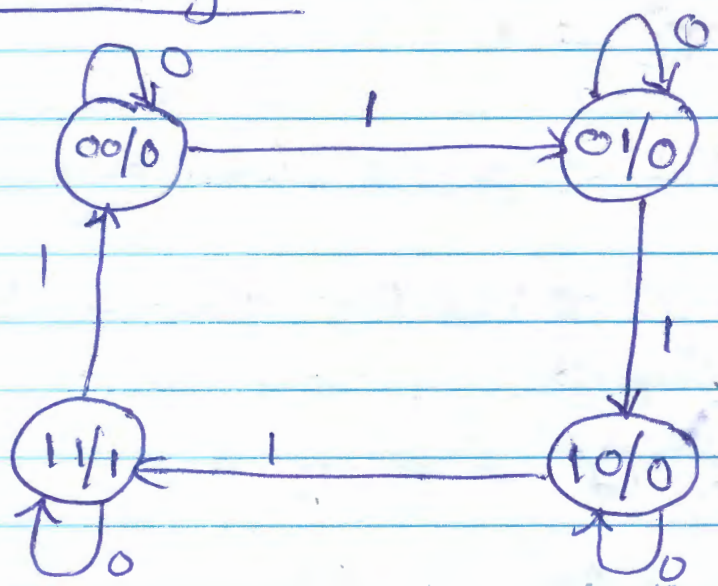
$= AB' + Ax' + A'Bx$

& $B(t+1) = x \oplus B$

## State Table

| Present state | | Input | Next State | | Output |
|---|---|---|---|---|---|
| A | B | x | A | B | Y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |

## state diagram



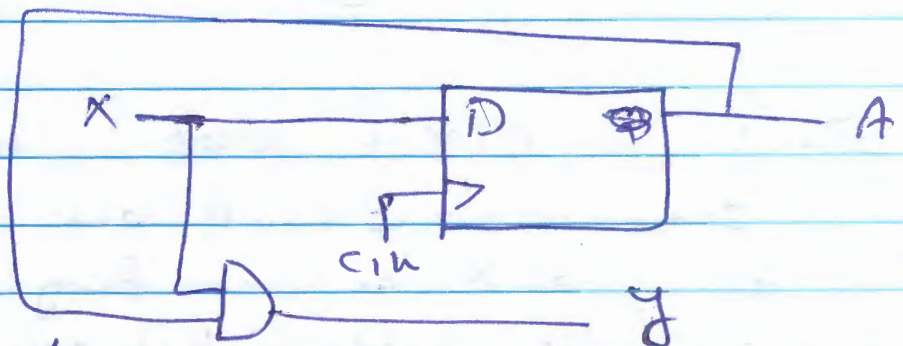
# Mealy and Moore Models (Machines).

- Sequential circuits are called Finite State Machine (FSM).

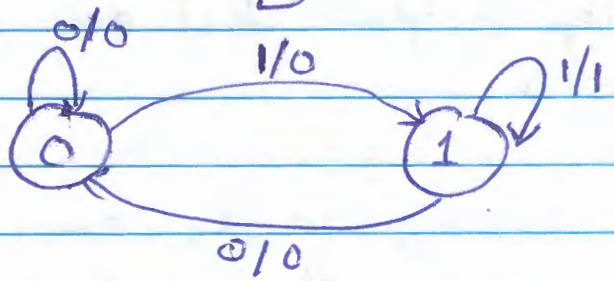
- In the Mealy Model (Mealy FSM) the output is a function of both the present state and input.

- In the Moore Model (Moore FSM) the output is a function of of the present state only.

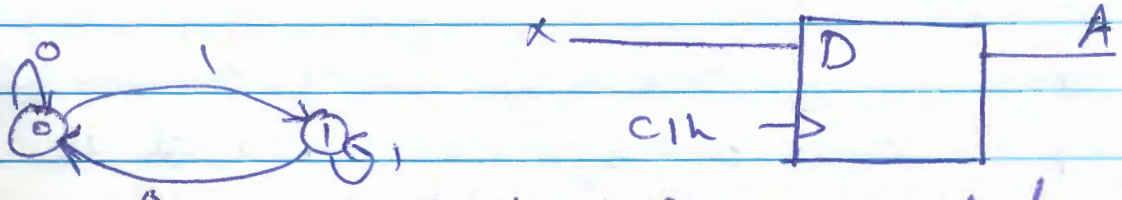
Ex.




y is function of present state & input ⇒ Mealy Model




Ex.

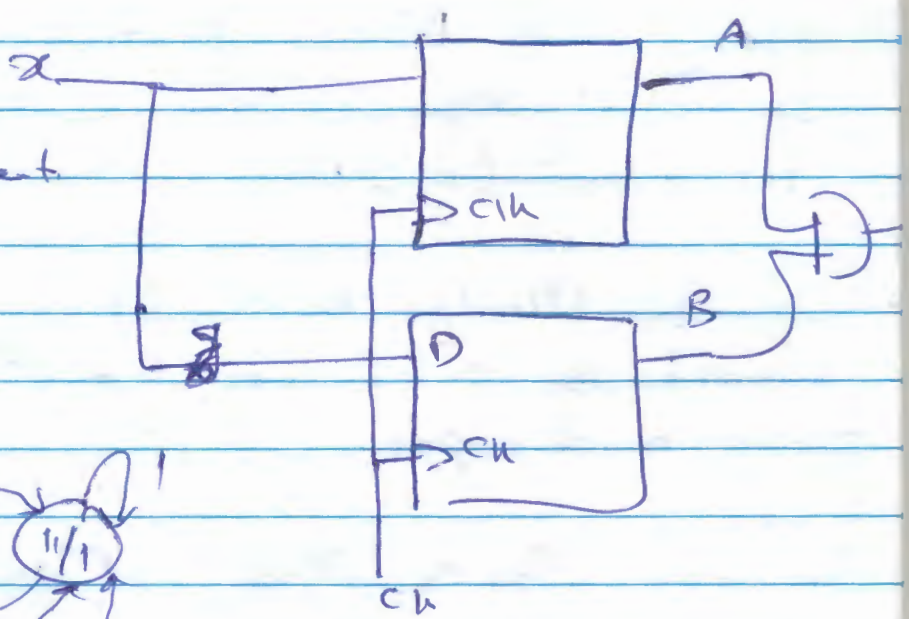


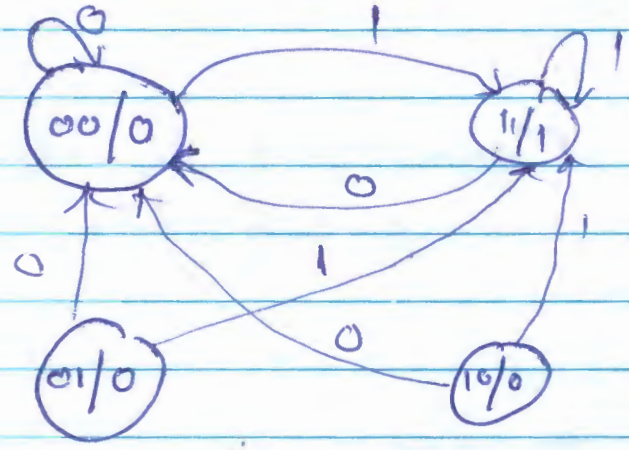

A (state itself is considered as ~~function~~ output).

⇒ Moore FSM

Ex:

y outputs is function of present state only
⟹ Moore Machine




(*) - In a Moore model, the outputs of the sequential circuit are synchronized with the clock because they depend on only flip-flop outputs that are synchronized with the clock.

- In a Mealy mode, the outputs may change if the inputs change during the clock cycle

- In order to synchronize Mealy machines ⟹ inputs must be synchronized with the clock and outputs sampled only during the clock edge.

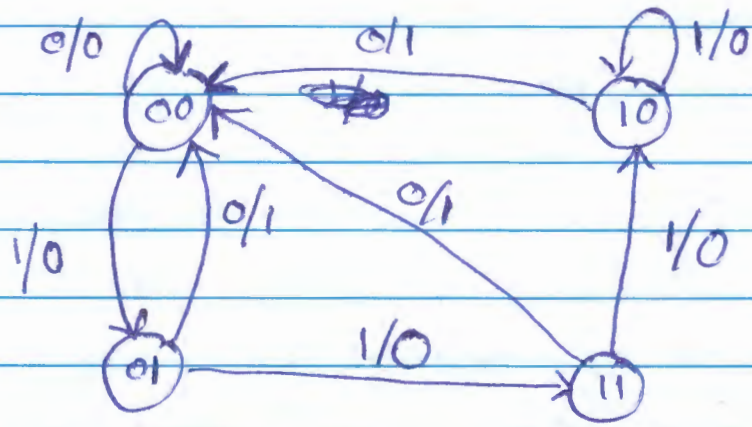
## Structural Description of Sequential circuit

fig 5-20 (a)

```
module T_FF (Q, T, clk, Reset);
  output Q;
  input T, clk, Reset;
  reg Q;
  always @ (posedge clk or negedge Reset)
    if (Reset == 0) Q = 1'b0;  // if(~reset)
    else Q = Q ^ T;
endmodule
```

```
module Tcircuit (x, y, A, B, clk, reset);
  input x, clk, reset;
  output y, A, B;
  wire TA, TB;
  assign TB = x;
  assign TA = x & B;
  assign y = A & B;
  T_FF FFA (A, TA, clk, Reset);
  T_FF FFB (B, TB, clk, Reset);
endmodule
```

(*) Fig. 5-16 state Diagram:




```
module Mealy (x, y, clk, ~~reset~~, A, B);
input x, clk, ~~reset~~;
output y, A, B;
reg y, A, B;
Parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10,
          S3 = 2'b11;
always @ (posedge clk) ~~or negedge reset~~
~~if (~reset) {A, B} = S0;~~
~~else~~
case ({A, B})
  S0 : if (x) {A, B} = S1;
       else   {A, B} = S0;
  S1 : if (x) {A, B} = S3;
       else   {A, B} = S0;
  S2 : if (x) {A, B} = S2;
       else   {A, B} = S0;
  S3 : if (x) {A, B} = S2;
       else   {A, B} = S0;
endCase
```

```
always @ (A or B or x)
  case ({A, B})
    S0 : y = 0;
    S1 : if (x) y = 1'b0; else y = 1'b1;
    S2 : if (x) y = 1'b0; else y = 1'b1;
    S3 : if (x) y = 1'b0; else y = 1'b1;
  endcase
endmodule
```

# STATE REDUCTION AND ASSIGNMENT

- The reduction of the number of flip-flops in a sequential circuit is referred to as the state reduction.

- In the following examples, only the input-output sequences are important (not the state itself).

- Example of Fig- 5-22 (State Diagram)
  - There are an infinite number of input sequences that may be applied to the circuit, each sequence results in a unique output sequence.
  - Consider an input sequence 0101011010 0 starting from the initial state a.

| state | a | a | b | c | d | e | f | f | g | f | g | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| input | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | |
| output | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | |

- Now we want to reduce the number of states such that the same input sequences will produce the same output sequences
- It is easier to use the state table for this purpose.

(*)

(*) state table

| Present State | Next state x=0 | Next state x=1 | output x=0 | output x=1 |
|---|---|---|---|---|
| a | a | b | 0 | 0 |
| b | c | d | 0 | 0 |
| c | a | d | 0 | 0 |
| d | e | f | 0 | 1 |
| e | a | f | 0 | 1 |
| f | g | f | 0 | 1 |
| g | a | f | 0 | 1 |

(*) The Rule for state reduction

" Two states are said to be equivalent if, for each member of the set of inputs, they give exactly the same output and send the circuit either to the same state or to an equivalent state".

⇒ (1) state e = g (equivalent to) ⇒ new state table

| Present State | Next state x=0 | Next state x=1 | output x=0 | output x=1 |
|---|---|---|---|---|
| a | a | b | 0 | 0 |
| b | c | d | 0 | 0 |
| c | a | d | 0 | 0 |
| d | e | f | 0 | 1 |
| e | a | f | 0 | 1 |
| f | e | f | 0 | 1 |

(2) state d = f ⇒ new state table

| Present state | Next State x=0 | Next State x=1 | output x=0 | output x=1 |
|---|---|---|---|---|
| a | a | b | 0 | 0 |
| b | c | d | 0 | 0 |
| c | a | d | 0 | 0 |
| d | e | d | 0 | 1 |
| e | a | d | 0 | 1 |

⇒ no remaining equivalent state.

⇒ new & final state diagram (Fig 5-23)

(*) Now for the same sequence of inputs:-

0 1 0 1 0 1 1 0 1 0 0 ⇒

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| State | a | a | b | c | d | e | d | d | e | d | e | a |
| input | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | |
| output | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | |

(*) State Assignment

- After state reduction, we assign each of the remaining states a unique binary code as follows:-

| State | Assignment 1 (binary) | Assignment 2 (gray) | Assignment 3 (one-hot) |
|---|---|---|---|
| a | 000 | 000 | 00001 |
| b | 001 | 001 | 00010 |
| c | 010 | 011 | 00100 |
| d | 011 | 010 | 01000 |
| e | 100 | 110 | 10000 |

↓ unused

amh1@hotm

⇒ final state table

| Present State | Next State x=0 | Next State x=1 | output x=0 | output x=1 |
|---|---|---|---|---|
| 000 | 000 | 001 | 0 | 0 |
| 001 | 010 | 011 | 0 | 0 |
| 010 | 000 | 011 | 0 | 0 |
| 011 | 100 | 011 | 0 | 1 |
| 100 | 000 | 011 | 0 | 1 |

Example:

State Table

| Present state | Next State x=0 | Next State x=1 | output x=0 | output x=1 |
|---|---|---|---|---|
| a | c | b | 0 | 1 |
| b | d | a | 0 | 1 |
| c | a | d | 1 | 0 |
| d | b | d | 1 | 0 |

$a \neq c$, $a \neq d$, $b \neq c$, $b \neq d$ (different outputs)

$a = b$ if $c = d \Rightarrow (a,b)$ imply $(c,d)$

also $c = d$ if $a = b \Rightarrow (c,d)$ imply $(a,b)$

⟹ $a = b$ & $c = d \Rightarrow$ new state table

| Present State | Next State x=0 | Next State x=1 | output x=0 | output x=1 |
|---|---|---|---|---|
| a | c | a | 0 | 1 |
| c | d | c | 1 | 0 |

* Two states are equivalent if for each possible input, they give exactly the same output and go to the same next state or to equivalent next states.

* Implication Table:
a systematic way to reduce the number of states.

Ex: State Table

| Present State | Next State | | Output | |
|---|---|---|---|---|
| | x=0 | x=1 | x=0 | x=1 |
| a | d | a | 0 | 0 |
| b | e | a | 0 | 0 |
| c | g | f | 0 | 1 |
| d | a | d | 1 | 0 |
| e | a | d | 1 | 0 |
| f | c | b | 0 | 0 |
| g | a | e | 1 | 0 |