Sobre o circuito da UC

Para implementar o controle da UC sobre o processador, definiremos uma interface composta de flags. Definimos flag como um "bit de estado" (status bit), que indica como determinado aspecto do processador se comporta ou como se encontra.

A interface deve ser feita de modo que cada flag seja simples o suficiente para ser implementada no circuito com um bit, e o conjunto das flags deve permitir controle o suficiente sobre o processador para que a UC possa fazer executar todas as instruções da especificação. Assim, definimos-nas:

Flags

0.	RAM → IR
1	IR → RAM
2	RAM → AC
3	AC → RAM
4	PC → RAM
5	RAM → PC (incond.)
6	RAM → PC if >
7	RAM → PC if =
8	INPUT → RAM
9	RAM → output
10	RAM [write / read]
11	RAM → ULA B
12	0x00 → ULA B
13	ULA R → AC
14	ULA OP [+ / −]
15	reset & stop PC

tabela 1 - flags

- Na tabela ao lado, a ocorrência de uma seta (→) indica que a flag determina se há conexão entre os elementos que a contam e são apontados por ela. Se a flag estiver em zero não há conexão, se estiver em um, há conexão
- Quando houver colchetes, leia-se [flag = 0 / flag = 1]
- A flag 15 só reseta o PC quando está em 1.

Assim, definida a interface, podemos calcular quais flags devem ser "setadas" dependendo de cada instrução. Por exemplo, se a instrução é "NOP", todas as flags devem estar em 0. Se a instrução é "copie do endereço EE para o AC", as flags "IR → RAM", "RAM → AC" e "RAM [w/r]" devem estar setadas. Já que cada instrução e cada flag corresponde a um número, temos uma função Booleana

$$p: 2^{4} \rightarrow 2^{16}$$

Descrever a tabela verdade desta função p é determinar o funcionamento do processador a partir da interface. É isso que fazemos a seguir:

Descrição da instrução	Código da instrução c	p(c) (Ver explicação sobre a notação)
NOP	0x00	[]
COPIE [EE] para o AC	0x01	[1, 2, 10]
COPIE [AC] para o end. EE	0x02	[1, 3]
Some [EE] com [AC]	0x03	[1, 10, 11, 13]
Subtraia [EE] de [AC]	0x04	[1, 10, 11, 13, 14]
Leia um número e guarde-o no end. EE	0x07	[1, 8]
Imprima [EE]	0x08	[1, 9, 10]
Pare	0x09	[15]
Desvie para EE	0x0A	[5]
Desvie para EE se AC>0	0x0B	[6, 12]
Desvie para EE se AC=0	0x0D	[7, 12]

tabela 2 - função p

Na tabela acima, usamos a seguinte notação:
Para $a_i, b_j, i, j, n \in \mathbb{N}$, definimos que

$$[a_0, a_1, \dots, a_n] \overset{\text{def}}{=} (b_0, b_1, \dots, b_{15}) \overset{\text{def}}{=} t$$

é a tupla tal que, se b_j é o (a_i+1)-ésimo elemento de t, para algum i e algum j, então $b_j = 1$. Caso contrário, $b_j = \emptyset$.

Em outras palavras, se para alguma instrução c vale

$$p(c) = \{a_0, a_1, \dots a_n\},$$

Então essa instrução deve fazer as flags a_i tornarem-se 1 e as demais tornarem-se $\emptyset$.

Sob outra ainda perspectiva, se se considerar que uma tupla t como acima, de 16 elementos, pode representar um número em base 2, então

$$[a_0, a_1, \dots, a_n] = 2^{a_0} + 2^{a_i} + \dots + 2^{a_n}$$

Com a função p, a UC fica bem determinada. Mas para desenhar seu circuito, convém definir as funções $f_i : 2^4 \to 2$, $i \in \{0, 1, \dots, 15\}$, tais que

$$p(c) = (f_0(c), f_1(c), f_2(c), \dots, f_{15}(c))$$

De modo que f_i determina o comportamento da flag i.
A partir da tabela 2, determinamos f_i

- $f_0(c) = \emptyset$
- $f_1(c) = \sum m(1,2,3,4,7,8)$
- $f_2(c) = \sum m(1)$
- $f_3(c) = \sum m(2)$
- $f_4(c) = \not\emptyset$
- $f_5(c) = \sum m(10)$
- $f_6(c) = \sum m(11)$
- $f_7(c) = \sum m(13)$
- $f_8(c) = \sum m(7)$
- $f_9(c) = \sum m(8)$
- $f_{10}(c) = \sum m(1,3,4,8)$
- $f_{11}(c) = \sum m(3,4)$
- $f_{12}(c) = \sum m(11,13)$
- $f_{13}(c) = \sum m(3,4)$
- $f_{14}(c) = \sum m(4)$
- $f_{15}(c) = \sum m(9)$

A essa altura, não está claro por que há flags que sempre estão em $\emptyset$, como a $\emptyset$ e a 4. Acontece que a função acima descrita dá conta apenas da etapa "execute" do ciclo FDX. Além disso, o processador deve poder fazer acontecer a etapa "fetch". Assim, definimos a função $P: 2^5 \to 2^{16}$

$$P(c, Fx) = (g_0(c, Fx), g_1(c, Fx), \dots, g_{15}(c, Fx))$$

Onde Fx é um bit que diz se a etapa atual é de "Fetch", quando $Fx = \emptyset$ ou "execute", quando $Fx = 1$.

Já que na etapa fetch as flags setadas devem ser $\emptyset$, 4 e 10, vale o seguinte:

Para $i \in \{0, 4, 10\}$, $g_i(c, Fx) = f_i(c) + \overline{Fx}$

Para $i \in \{0, 1, \dots, 15\} \setminus \{0, 4, 10\}$, $g_i(c, Fx) = f_i(c) \cdot Fx$

Por fim, podemos montar o circuito da UC