STUDIA UNIV. BABEŞ-BOLYAI, MATHEMATICA, XL, 3, 1995

AN AXIOMATIC APPROACH TO ASSEMBLERS

Zoltán CSÖRNYEI*

Received: 30.07.1995
AMS subject classification: 68N05, 68N15, 68N20

REZUMAT. - O abordare axiomatică a asamblоarelor. Scopul principal al asamblоarelor este de a atribui valori simbolurilor din program. Programul obiect se asamblează din aceste valori. Se studiază procesul de atribuire a valorilor, valoarea simbolurilor se generează pe baza numărului de treceri şi a gradului de postdefinitate. Se determină numărul exact al trecerilor necesare traducerii programului de asamblare.

Abstract. - The main purpose of assemblers is to assign values to the symbols of an assembly program. The target program is assembled from these values. The process of assigning values is studied, the values of the symbols are generated depending on the number of passes and the degree of postdefinity. The exact number of passes needed to translate an assembly program is determined.

1. Introduction. The assembler creates the target program and the list from the assembly language program. The assembly language program is a series of symbols, during the translation the assembler assigns values to these symbols, and the target program and the list are composed from these values.

Emphasizing the common characteristics of assembly languages [1], [2], [7]-[11] firstly we define a grammar and the assembly language examined is generated by this grammar. The

* *Eötvös Loránd University, Department of General Computer Science, Budapest Hungary, E-mail: csz@maxi.inf.elte.hu*

symbols of the assembly language programs are classified depending on their positions. The syntactic error-free assembly language programs will be given by axioms, and The Fundamental Axiom of Assemblers [12] will be stated. In order to study the process of assigning values, the values of symbols are defined precisely and the classes of pre- and post-definit symbols are determined.

The degree of postdefinity [4] is introduced, and the relation between the value and the degree of postdefinity is studied to determine the minimal number of passes needs to translate the assembly program [3]. The result of this study is summarized in The Fundamental Theorem of Assemblers [5].

2. The assembly language. To study the assembly languages we define a simple context-free grammar G, and the assembly language program is a sentence of language $L(G)$ generated by grammar G. We give only the first few productions:

(1) <*program*> → <*line*> eol |
<*program*><*line*> eol
(2) <*line*> → <*symb*> = <*expr*> |
<*stmt*>
(3) <*stmt*> → <*symb*> : <*stmt*> |
<*mnem*> |
<*mnem*><*opnd*>
(4) <*opnd*> → <*expr*> |
<*opnd*> , <*expr*>
(5) <*mnem*> → ADD|MOV|...
....

From the above productions it is obvious that the unit of the assembly language program is the *line*, and that the line can be decomposed into three fields: label, instruction and operand field. The instruction field contains either a *directive* (e.g.=) or a *mnemonic* (e.g. ADD, MOV).

We complete the line with the serial number field. Thus, the k-th line of the assembly

language program has the following structure:

(0)	(1)	(2)	(3)
k	*label*	*statement*	*operand*

With the production (3), it is possible to give labels in the program, and it can be seen that it is possible to have more labels for one mnemonic:

$symb_1$: $symb_2$: ... $symb_m$: *mnemonic operand* eol

The assembler assigns the actual value of the *current location counter* to the label. Marking this counter by \$, we transform the above line into the following form:

$symb_1$ = \$ eol
$symb_2$ = \$ eol
...
$symb_m$ = \$ eol
mnemonic operand eol

After the transformation the assembly language program *P* consists of rows of the next types:

(0)	(1)	(2)	(3)
k		*mnemonic*	*operand*

k		*mnemonic*	

k	*symbol*	=	*operand*

Let $a(k)$, $m(k)$ and $u(k)$ denote the set of symbols in the field (1), field (2) and field (3) of the k-th line, respectively, and let $|P| = \max k$. Now we define the set of symbols:

DEFINITION 2.1 *Let* $A(0) = \{\$\}$, $A(k) = A(k\text{-}1) \cup a(k)$ $(1 \leq k \leq |P|)$, *and* $A = A(|P|)$. *Set A is called the parameter set of program P.*

We note that $\$ \notin a(k)$ $(1 \leq k \leq |P|)$.

Those $a \in A$ parameters which appear in the label fields in the line

(0)	(1)	(2)	(3)
k	a	=	$

are called *labels,* and let us denote by $b(k)$ the label of the k-th line:

$b(k) = \{a(k) \mid m(k) = \{=\} \wedge u(k) = \{\$\}\}$

The set of labels is the following:

DEFINITION 2.2 *Let* $B(0) = \varnothing$, $B(k) = B(k\text{-}1) \cup b(k)$ $(1 \leq k \leq |P|)$, *and* $B = B(|P|)$. *Set B is called the* label set *of program P.*

In the assembly language program the labels need to be unique, that is a label cannot appear in other label fields of the program. This is stated by the following axiom.

Axiom I $b(k) \notin B(k\text{-}1)$, $1 \leq k \leq |P|$

If a label does not satisfy this axiom, then the assembler reports a "multiple defined symbol" error on every occurence of this label.

In the instruction fields there are directives or mnemonics:

DEFINITION 2.3 *Let* $M(0) = \varnothing$, $M(k) = M(k\text{-}1) \cup m(k)$ $(1 \leq k \leq |P|)$, *and* $M = M(|P|)$. *Set M is called the* instruction symbol set *of program P.*

Into the instruction field only an element of a previously given set $\mathcal{M}$ can be written. The set $\mathcal{M}$ is called the *instruction set of the assembly language.* The requirements for this set are stated by the following axioms:

Axiom II $M \subseteq \mathcal{M}$

Axiom III $A \cap M = \varnothing$

The symbols found in the operand fields are given in the following way:

DEFINITION 2.4 *Let* $U(0) = \varnothing$, $U(k) = U(k\text{-}1) \cup u(k)$ $(1 \leq k \leq |P|)$, *and* $U = U(|P|)$. *Set U is called the* operand symbol set *of the program P.*

The $u(k)$ consists constants and operators, too, denote $u_C(k)$ and $u_O(k)$ the set of constants and the set of operators in the k-th line, and let U_C, U_O denote the set of constants and the set of operators of program P.

Similarly to the instructions, the elements of U_O can only be elements of a previously given set $\mathcal{O}$, and the restrictions for these sets are stated by the following axioms:

Axiom IV $U_O \subseteq \mathcal{O}$

Axiom V $U_O \cap A = \varnothing \wedge U_O \cap M = \varnothing$, *and*

$$U_C \cap A = \varnothing \wedge U_C \cap M = \varnothing$$

The non-operator and non-constant symbols found in the operand field must be defined in the label field, that is if

$$U_A = U \setminus (U_C \cup U_O),$$

then every element of U_A must be a parameter. This is stated in the fundamental axiom of assemblers.

Axiom VI (The Fundamental Axiom of Assemblers) $U_A \subseteq A$

Let $u_A(k) = u(k) \setminus (u_C(k) \cup u_O(k))$ $(1 \leq k \leq |P|)$. If for a symbol $s \in u_A(k)$ the above axiom does not hold, then the assembler in the k-th line gives an *"unknown symbol"* error message for the ymbol s.

3. The values of symbols. We will now examine what kind of values the assembler assigns to the elements of the previously defined sets of symbols. Let $val(s,k)$ denote the value of symbol s in line k. If the assembler cannot determine this value, then let it be $val(s,k) = \varepsilon$.

Axiom VII *For every symbol* $s \in U_C \cup M$, $1 \leq k \leq |P|$ *the inequality* $\mathrm{val}(s,k) \neq \varepsilon$, *and for every* $1 \leq k, l \leq |P|$ *the equality* $\mathrm{val}(s,k) = \mathrm{val}(s,l)$ *is valid.*

The above axiom states that the values of constants are determined unambiguously, and these values are independent from the lines. On the basis of the axiom the same is true for the mnemonic, that is the machine code assigned to the statement is independent from the program.

We only have to deal with the elements of sets A and U_O. As $\$ \in A$, we examine the value of this symbol first.

DEFINITION 3.1 *Let* val(\$,1) = 0, *and for* $k \geq 2$ *let*

$$\text{val}(\$,k) = \text{val}(\$,k\text{-}1) + f(m(k\text{-}1),u(k\text{-}1)),$$

where $f(m,u)$ *is a non-negative integer function which gives the length of the machine code, its value for every* $m(k\text{-}1) \subseteq M$ *and for every* $u(k\text{-}1) \subseteq U$ *is previously given, even in the case of* $s \in u(k\text{-}1)$, $\text{val}(s,k\text{-}1) = \varepsilon$.

According to the definition the value of the current location counter is zero in the first line, and the value of the increase is determined by the mnemonic and the symbols of the operand field, and not by the values of these symbols.

Let $A' = A \setminus \{\$\}$, $U_A' = U_A \setminus \{\$\}$, and let

$$u_A'(k) = \begin{cases} u_A(k) & \text{if } \$ \notin u_A(k) \\ u_A(k) \setminus \{\$\} & \text{if } \$ \in u_A(k) \end{cases}$$

In the following we deal with the symbols of A'. We can see that if $s \in A'$ is in the field (1), then in the operand field (3) only one expression can appear:

(0)	(1)	(2)	(3)
k	*s*	=	*expr*

The directive = means the assigning value, and for this reason we first give the value of the expresion.

We say that for an expression *expr* the inequality $val(expr,k) \neq \varepsilon$ is valid if

$u'_A(k) = \varnothing$ or if for all $s \in u'_A(k)$ the value of symbol s is known. Then

$$val\ (expr\ ,\ k) = \underset{u_0(k)}{\times}\ val\ (s,\ k),$$

where $s \in u_A(k) \cup u_C(k)$ and $\times$ means the execution of operations in $u_O(k)$.

If in the expression *expr* the $u'_A(k) = \varnothing$, then $val(expr,k) \neq \varepsilon$, as the expression at the very most can only contain symbol \$, constants and operators. The value of \$ according to Definition 3.1, the values of constants according to the Axiom VII are known.

A symbol s can occur in the label and operand field of the same line:

(0)	(1)	(2)	(3)
k	s	=	$s + 1$

and from this it can be seen that the value of the ymbol s in the two fields will not be the same. Let $val\ (s,\ k^{(j)})$ denote the value of symbol s in field j of the k-th line.

If $s \in a(k)$, then let $val\ (s,\ k^{(1)}) = val\ (expr\ ,\ k)$. We note if $u'_A(k) = \varnothing$, then $val\ (s,\ k^{(1)}) \neq \varepsilon$. If $val(expr,k) = \varepsilon$, then $val(s,k^{(1)}) = \varepsilon$.

As a symbol can appear in the label fields of different lines, we define the *scope* of symbols.

DEFINITION 3.2 *If* $s \in a(k_1),\ s \in a(k_2),\ \ldots,\ s \in a(k_m)$, *and* $1 \leq k_1 < k_2 < \ldots < k_m \leq |P|$, *then let the scope of symbol* s $(s \in a(k_i))$ *be*

$$B(s,\ k_i) = \{(k_i + 1)^{(3)},\ (k_i + 2)^{(3)},\ \ldots,\ k_{i+1}^{(3)}\},\ \textit{if } i \neq m, \textit{ and}$$

$$B(s,\ k_m) = \{1^{(3)},\ 2^{(3)},\ \ldots,\ k_1^{(3)},\ (k_m+1)^{(3)},\ \ldots,\ |P|^{(3)}\}.$$

We note that for $s \in b(k)$ according to Axiom I

$$B(s,\ k) = \{1^{(3)},\ 2^{(3)},\ \ldots,\ |P|^{(3)}\}.$$

When we assign values to symbols, a very important property is the pre and postdefinity of symbols.

DEFINITION 3.3 *If* $s \in a(k)$ *and* $l^{(3)} \in E(s,k)$, *then symbol* s *in field* $l^{(3)}$ *in the case of*

$k < l$ *is said to be* predefinite, *in the case of* $k \geq l$ *is said to be* postdefinite *symbol.*

If $s \in a(k_1)$, $s \in a(k_2)$, ..., $s \in a(k_m)$, then symbol s in fields $i^{(3)}$, $2^{(3)}$, ..., $k_1^{(3)}$ is a postdefinite, and in fields $(k_1+1)^{(3)}$, ..., $|P|^{(3)}$ is a postdefinite, and in fields $(k_1+1)^{(3)}$, ..., $|P|^{(3)}$ is a predefinite symbol. Similarly, if label $s \in b(k)$, then label s in fields $1^{(3)}$, $2^{(3)}$, ..., $k^{(3)}$ is a postdefinite, and in fields $(k+1)^{(3)}$, ..., $|P|^{(3)}$ is a predefinite symbol.

From the last example, in which symbol s is in the label and the operand field, it is obvious that the value of the symbol s is depending on not only the fields but the passes, too. We extend the definition of values for passes, let $val^{(i)}(s, k^{(j)})$ denote the value of symbol s at pass i, in field j of the k-th line of the program.

For symbol \$ let

$$val^{(i)}(\$, k^{(3)}) = val(\$, k) \quad (1 \leq k \leq |P|)$$

where $val(\$,k)$ is the value given in Definition 3.1. At the begining of translation the value of symbol $s \in A'$ is unknown, that is for all $s \in A'$

$$val^{(1)}(s, 1^{(3)}) = \varepsilon,$$

and symbol s can only get value in such a first line, where in the label field appears:

DEFINITION 3.4 *If symbol* $s \in a(k)$ *and* $i \geq 1$, *then,*

$$\text{val}^{(i)}(s, k^{(1)}) = \begin{cases} \varepsilon & \text{if } \exists\, r \in u_A(k), \\ & \text{and } \text{val}^{(i)}(r, k^{(3)}) = \varepsilon \\ \underset{u_0(k)}{\times} \text{val}^{(1)}(q, k^{(3)}) & \text{otherwise} \end{cases}$$

where $q \in u_A(k) \cup u_C(k)$, *and* $\times$ *means the execution of operation in* $u_O(k) \subseteq U_O$.

If we determine the value of a symbol, then the value in all of predefinite references is equal to this value, and the postdefinite references only get their values in the next pass.

DEFINITION 3.5 *If* $s \in a(k)$, $s \in u'_A(l)$, $l^{(3)} \in B(s, k)$ *and* $i \geq 1$, *then*

$\text{val}^{(i)}(s, l^{(3)}) = \text{val}^{(i)}(s, k^{(1)})$, *if* $k < l$, *and*

$\text{val}^{(i+1)}(s, l^{(3)}) = \text{val}^{(i)}(s, k^{(1)})$, *if* $k \geq l$.

From these definitions unfortunately it can not be seen that the value of a symbol is determined in which pass. This is why we introduce the degree of postdefinity, and the relation between the value and the degree of postdefinity will be investigated.

4. The degree of postdefinity. The value of a parameter in the label field is undefined if in the operand field of the same line there is a symbol with undefined value. It is possible that the assembler gives a value to this symbol of the operand field in one of passes, and therefore in this pass and in the next passes the value of the parameter in the label field can be determined.

DEFINITION 4.1 *If* $s \in a(k)$, *then the* degree of postdefinity *of symbol* s *in* $k^{(1)}$ *let*

$$\text{pd}(s, k^{(1)}) = \begin{cases} 0 & \text{if } u'_A(k) = \varnothing \\ \max\limits_{r \in u'_A(k)} \text{pd}(r, k^{(3)}) & \text{otherwise} \end{cases}$$

According to the definition the symbol in the label field takes the highest degree of postdefinity from the operand field.

DEFINITION 4.2 *If* $s \in a(k)$, $s \in u'_A(l)$ *and* $l^{(3)} \in E(s, k)$, *then the degree* of postdefinity *of symbol* s *in* $l^{(3)}$ *let*

$$\text{pd}(s, l^{(3)}) = \begin{cases} \text{pd}(s, k^{(1)}) & \text{if } l > k \\ \text{pd}(s, k^{(1)}) + 1 & \text{if } l \leq k \end{cases}$$

The definition says that in postdefinite references the degree of postdefinity is one greater than in the line of its definition, and in predefinite refences the degree of postdefinity does not change.

If $s \in b(k)$, then $pd(s,k^{(1)}) = 0$, that is the degree of postdefinity in the label field of its definition line is zero, and according to the definitions, the degree of postdefinity in the predinite references equals zero and in the postdefinite references it equals one.

THEOREM 4.1 *If* $s \in a(k)$, *then* $\mathrm{pd}(s,k^{(1)}) \geq 0$.

Proof. The statement is a consequence of Definition 4.1 and 4.2. □

COROLLARY 4.1 *From Theorem* 4.1 *and Definition* 4.2 *it follows that* $\mathrm{pd}(s,1^{(3)})>0$ *for all symbols* $s \in u'_A(1)$.

THEOREM 4.2 *If* $\mathrm{pd}(s, l_1^{(3)}) \neq \mathrm{pd}(s, l_2^{(3)})$, *where* $1 \leq l_1 < l_2 \leq |P|$, *then* $\exists k$, *for which* $l_1 \leq k < l_2$ *and* $s \in a(k)$.

Proof. If $l_1^{(3)}, l_2^{(3)} \in E(s, k)$, where $s \in a(k)$, then the theorem is a consequence of Definitions 4.1 and 4.2.

If $l_1^{(3)} \in E(s, k_1)$, and $l_2^{(3)} \in E(s, k_2)$, where $s \in a(k_1)$, $s \in a(k_2)$ and $k_1 \neq k_2$, then k_1, k_2 can not be elements of neither set $E_1 = \{1, ..., l_1-1\}$, nor set $E_2 = \{l_1, ..., l_2-1\}$, nor set $E_3 = \{l_2, ..., |P|\}$ at the same time, otherwise the inequality $pd(s, l_1^{(3)}) \neq pd(s, l_2^{(3)})$ is not hold. Similarly, the case of $k_1 \in E_1$ and $k_2 \in E_3$ is not hold. If $k_1 \in E_1$ and $k_2 \in E_2$, or $k_1 \in E_2$ and $k_2 \in E_3$, then the statement is true, since in the first case let $k = k_2$, and in the second case let $k = k_1$. □

Clearly not every symbol has a finite degree of postdefinity. For example, if symbol s only appears in the line

(0)	(1)	(2)	(3)
k	s	=	$s + 1$

then according to Definitions 4.1 and 4.2

$$pd(s, k^{(1)}) = pd(s, k^{(3)}) = pd(s, k^{(1)}) + 1,$$

and we can state the following theorem:

THEOREM 4.3. *If* $s \in u'_A(k)$ *and* $k^{(3)} \in E(s, k)$, *then* $\mathrm{pd}(s, k^{(1)}) = \infty$

If $s \in u'_A(1)$, then Corollary 4.1 shows that $pd(s, 1^{(3)}) \geq 1$. If $pd(s, 1^{(3)}) < \infty$, then according to Theorem 4.3 $k \neq 1$ is true in the case of $1^{(3)} \in E(s,k)$,

that is $\exists k$ $(1 < k \leq |P|)$, for which $1^{(3)} \in E(s,k)$. Thus, using Definition 4.2, we can state the following corollary:

COROLLARY 4.2 *If* $s \in u'_A(1)$ *and* $\mathrm{pd}(s, 1^{(3)}) = n < \infty$, *then in the case of* $1^{(3)} \in E(s,k)$ *it is true that* $k \neq 1$ *and* $\mathrm{pd}(s,k^{(1)}) = n\text{-}1$.

There is an obvious generalization of Theorem 4.3:

THEOREM 4.4 *If for symbols* $s_1, s_2, \ldots, s_m$

$s_1 \in a(k_1)$, $s_2 \in u'_A(k_1)$, $k_1^{(3)} \in E(s_2, k_2)$,

$s_1 \in a(k_2)$, $s_3 \in u'_A(k_2)$, $k_2^{(3)} \in E(s_3, k_3)$,

...

$s_m \in a(k_m)$, $s_1 \in u'_A(k_m)$, $k_m^{(3)} \in E(s_1, k_1)$,

then $\mathrm{pd}(s_i, k_i^{(1)}) = \infty$ $(1 \leq i \leq m)$.

Proof. The theorem follows from Definitions 4.1 and 4.2. □

We note that the infinite degree of postdefinity is based on a 'circle' in the declarations of symbols.

The next theorem states that if in the assembly language program there is a symbol with finite degree of postdefinity, then there is a symbol with zero degree of postdefinity too.

THEOREM 4.5 *If* $s \in a(k)$ *and* $\mathrm{pd}(s, k^{(1)}) < \infty$, *then* $\exists r \in A'$ *and* $\exists l$ $(1 \leq l \leq |P|)$, *for which* $r \in a(l)$ *and* $\mathrm{pd}(r,l^{(1)}) = 0$.

Proof. Let $pd(s, k^{(1)}) = i$, according to Theorem 4.1 $0 \leq i \leq \infty$. We prove the theorem by induction.

If $i = 0$, then let $r = s$, and $l = k$. Let us now suppose that the statement is true for every $j \leq i\text{-}1$, and we prove that the statement is true for $j = i$, too.

If $pd(s, k^{(1)}) = i$, then Definition 4.1 shows that $\exists r_1 \in u'_A(k)$, for which $pd(r_1, k^{(3)}) = i$, and if $r_1 \in a(k_1)$, then $k^{(3)} \in E(r_1,k_1)$. According to Theorem 4.3 $k_1 \neq k$. If k_1

$> k$, then according to Definition 4.2 $pd(r_1, k_1^{(1)}) = i - 1$, and due to the hypothesis of the induction the statement is true.

If $k_1 < k$, then by Definition 4.1 it follows that $\exists r_2 \in u_A'(k_1)$, for which $pd(r_2, k_1^{(3)}) = i$, and if $r_2 \in a(k_2)$, then $k_1^{(3)} \in E(r_2, k_2)$. As $k_1 \neq k_2$, in the case of $k_2 > k_1$ the hypothesis of the induction is applicable, and if $k_2 < k_1$, then $\exists r_3 \in u_A'(k_2)$, for which $pd(r_3, k_2^{(3)}) = i$.

It can be seen that either we reach a symbol with degree of postdefinity i-1, or there exist

$$k > k_1 > \ldots > k_m$$

and $r_{m+1} \in u_A'(k_m)$, $pd(r_{m+1}, k_m^{(3)}) = i$. As $k_m \geq 1$, in the worst case $k_m = 1$. In this case however, due to Corollary 4.2, $pd(r_{m+1}, l^{(1)}) = i - 1$, and therefore the hypothesis of the induction is applicable. □

From the above proof it can be also seen that if there is a symbol for which the value of the degree of postdefinity is equal to n ($0 < n < \infty$), then there are symbols with degree of postdefinity 0,1,...,n-1:

COROLLARY 4.3 *If* $s \in a(k)$ *and* $\mathrm{pd}(s,k^{(1)}) = n$, *where* $0 < n < \infty$, *then* $\exists r_1, r_2, \ldots, r_{n-1} \in A'$, *and* $\exists l_1, l_2, \ldots, l_{n-1}$, *for which* $1 \leq l_j \leq |P|$, $r_j \in a(l_j)$, *and* $\mathrm{pd}(r_j, l_j^{(1)}) = j$ $(1 \leq j \leq n-1)$.

5. The relation between the value and the degree of postdefinity. We will now examine the relation between the value and the degree of postdefinity, and we will show that the degree of postdefinity of a symbol determines in which pass this symbol gets its value. Moreover, we prove that if this value was assigned in one of passes, then the value does not change in the following passes.

THEOREM 5.1 (The Fundamental Theorem of Assemblers)

If $s \in u'_A(l)$ *and* $pd(s, l^{(3)}) = n < \infty$ *then* $\text{val}^{(n+1)}(s, l^{(3)}) \neq \varepsilon$.

Proof. We prove the theorem by induction. In the case of $n = 0$ the method of proof of Theorem 4.5 can be used, and it will not be presented here. Suppose that for every symbol $s_i \in u'_A(k_i)$ in the case of $pd(s_i, k_i^{(3)}) = j - 1$ the statement is true, that is $val^{(j)}(s_i, k_i^{(3)}) \neq \varepsilon$.

We prove that if $s \in u'_A(l)$, $pd(s, l^{(3)}) = j$ is hold, then $val^{(j+1)}(s, l^{(3)}) \neq \varepsilon$.

If $s \in a(k)$ and $l^{(3)} \in E(s,k)$, then according to Theorem 4.3 $k \neq l$. If $k > l$, then using Definition 4.2 $pd(s, k^{(1)}) = j - 1$, and on the ground of the hypothesis of the induction $val^{(j)}(s, k^{(1)}) \neq \varepsilon$. Definition 3.5 states that $val^{(j+1)}(s, l^{(3)}) = val^{(j)}(s, k^{(1)})$, thus the statement of the theorem is valid in the case of $k > l$

If $k < l$, then by Definition 3.5 it follows that the equality $val^{(j+1)}(s, l^{(3)}) = val^{(j+1)}(s, k^{(1)})$ is hold. As according to Definition 4.2 $pd(s, k^{(1)}) = j$, and Definition 4.1 shows that $\exists s_1 \in u'_A(k)$, for which $pd(s_1, k^{(3)}) = j$. For this symbol s_1 $\exists k_1$, where $s_1 = a(k_1)$, $k^{(3)} \in E(s_1, k_1)$. If $k_1 > k$, then we have finished, according to Definitions 3.5 and 4.2 $val^{(j+1)}(s_1, k^{(3)}) = val^{(j)}(s_1, k_1^{(1)})$, and $pd(s_1, k_1^{(1)}) = j - 1$.

If $k_1 < k$, then repeating the above procedure for s and k, either we reach a symbol with degree of postdefinity j-1, or we get series $s, s_1, s_2, \ldots, s_n$ and $k > k_1 > k_2 \ldots > k_n$. In the worst case $k_n = 1$, and composeing the next element s_{n+1} of the series, we find that $pd(s_{n+1}, 1^{(3)}) = j$. According to the Definitions 3.5, using Corollary 4.2, in the case of $s_{n+1} \in a(k_{n+1})$ $pd(s_{n+1}, k_{n+1}^{(1)}) = j - 1$, and $val^{(j)}(s_{n+1}, k_{n+1}^{(1)}) = val^{(j+1)}(s_{n+1}, 1^{(3)})$. This value on the ground of the hypothesis

of the induction $\neq \varepsilon$, which completes the proof. □

Now we prove that if we assign a value to a symbol in the n+1-st pass, then this value does not change in the following passes.

THEOREM 5.2 *Let* $s \in u'_A(l)$. *If* $\mathrm{pd}(s, l^{(3)}) = n < \infty$ *then for all* $n_1, n_2 \geq n+1$ *the equality* $\mathrm{val}^{(n_1)}(s, l^{(3)}) = \mathrm{val}^{(n_2)}(s, l^{(3)})$ *is valid.*

Proof. We shall prove that $val^{(m)}(s, l^{(3)}) = val^{(m+1)}(s, l^{(3)})$ for every $m \geq n+1$, from this the statement of the theorem follows.

Suppose that the statement is not valid, that is the inequality $val^{(m)}(s, l^{(3)}) \neq val^{(m+1)}(s, l^{(3)})$ is true. We prove that from this condition it follows that there exists at least one symbol s_1, for which in the case of $s_1 \in a(k_1)$, $pd(s_1, k_1^{(1)}) \leq n-1$, and the values of symbol s_1 are different in two different passes.

Let $s \in a(k)$, $l^{(3)} \in E(s, k)$, then our condition is true according to Definition 3.5, if

1. in the case of $k > 1$ $val^{(m-1)}(s, k^{(1)}) \neq val^{(m)}(s, k^{(1)})$, but using Definition 4.2, $pd(s, k^{(1)}) = n-1$, so we have found a symbol which has different values in different passes, and its value of degree of postdefinity equals to n-1. Let $s_1 = s$ and $k_1 = k$.
2. in the case of $k < l$ $val^{(m)}(s, k^{(1)}) \neq val^{(m+1)}(s, k^{(1)})$, and according to Definition 4.2, $pd(s, k^{(1)}) = n$. Using Definitions 3.4, 4.1 and 4.2, in the worst case more times repeating the above line of reasoning and by Corollary 4.2 it follows that we surely reach a symbol which has different values in different passes, an its value of degree of postdefinity equals to n-1.

Therefore $\exists s_1 \in u'_A(k_1)$, for which $val^{(m-1)}(s_1, k_1^{(3)}) \neq val^{(m)}(s_1, k_1^{(3)})$ and $pd(s_1, k_1^{(3)}) \leq n-1$ in the case of m-1 ≥ n.

Using this method, we can produce a series of symbols $s_1, s_2, \ldots, s_{i_0}$ and a series of lines $k_1, k_2, \ldots, k_{i_0}$, for which $pd(s_{i_0}, k_{i_0}^{(1)}) = 0$, and $val^{(m)}(s_{i_0}, k_{i_0}^{(1)}) \neq val^{(m-1)}(s_{i_0}, k_{i_0}^{(1)})$. It is obvious that for the symbol s_{i_0}, with postdefinity degree zero, the above inequality is false, and thus we have proved the theorem. □

DEFINITION 5.1 *Let* pd(*P*) *denote the degree of postdefinity of program P*

$$\mathrm{pd}(P) = \begin{cases} 0 & \text{if } U_A' = \varnothing \\ \max\limits_{1 \le t \le |P|} \max\limits_{s \in u_A'(t)} \mathrm{pd}(s, k^{(3)}) & \text{otherwise} \end{cases}$$

The following theorem states that if the degree of postdefinity of a program is equal to n, then the program P can be translated by an n+1-pass assembler.

THEOREM 5.3 *For all symbols* $s \in U_A'$ *of program P in the case of* $s \in u_A'(l)$ $val^{(pd(P)+1)}(s, l^{(3)}) \neq \varepsilon$.

Proof. The statement is the consequence of Definitions 3.1, 3.4 and 3.5, as well as Theorems 5.1 and 5.2. □

As $U_A' \subset A$, according to the Definition 4.2 the statement of the above theorem is valid for every symbol $s \in A$.

COROLLARY 5.1 *For all symbols* $s \in A$ *of program P in the case of* $s \in a(k)$ $val^{(pd(P)+1)}(s, k^{(1)}) \neq \varepsilon$.

We have dealt with values of all symbols of an assembly program. We note that $pd(s, k^{(1)}) = 0$ in the case of $s \in b(k)$, and for $l^{(3)} \in E(s, k)$ $pd(s, l^{(3)}) \le 1$, the translation of labels can be done with a two-pass assembler, and this is the reason, that the most of assemblers has two passes. Thus, the most of assemblers unables to solve the translation of symbols of higher degree of postdefinity [6], and it is the programmers' responsibility to reduce the degree of postdefinity to 1, using modifications to the assembly language program, or simply with rearranging of lines.

REFERENCES

1. Brumm,P., Brumm,D., *8386/80486 Assembly Language Programming* (Windcrest/McGraw-Hill, 1993).
2. Chiu,P.P.K., Fu,S.T.K., A Generative Approacg to Universal Cross Assembler Design, *Sigplan Notices* Vol. 25(1990), No. 1. pp. 43-51.
3. Csörnyei Z.,: *Az assemblálás elmélete* (The Theory of Assemblers, in Hungarian) *Alkalmazott Matematikai Lapok* (to appear).
4. Csörnyei Z., On the Postdefinity of Symbols in Assembly Languages, *First Joint Conference on Modern Applied Mathematics,* Ilieni/Illyefava, Romania, 15-17 June, 1995. pp. 22-23.
5. Csörnyei Z., The Fundamental Theorem of Assemblers, *Conference at Dept. of Computer Science, Babeş-Bolyai University,* Cluj-Napoca, 8th November, 1994. (unpublished).
6. Csörnyei Z., "Univerzális mikroprocesszor assemblerek" (Universal microprocessor assemblers, in Hungarian), Doctoral Thesis Eötvös Loránd University, Budapest, 1980.
7. Giles,W.B., *Assembly Language Programming for the Intel 80xxx Family* (Macmillan Publishing Co., Inc., New York, NY., 1991).
8. Kindred,A.R., *Structured Assembler Language for IBM Microcomputers* (Hartcourt Brace, Orlando, FL., 1991).
9. Nelson,R.P., *Microsoft's 80386/80486 Programming Guide* (Microsoft Press. Redmond, WA., 1991).
10. Thorne,M., *Computer Organization and Assembly Language Programming for IBM PCs and Compatibles* (Benjamin-Cummings Publ. Co., Inc., Redwood City, CA., 1991).
11. Turley,J.L., *Advanced 80386 Programming Techniques* (Osborne McGraw-Hill, Berkeley, CA., 1988).
12. Wick,J.D., "Automatic generation of assemblers", Yale University, Dept. of Computer Science, Research Report No. 50., 1975.