Date :

Name : Aakash A. Joshi
Roll No : 0077
Branch : Computer Batch : T4
Subject : System Programming and Operating Systems

Topic : Assignment 1 (Theory)

Date :

Questions:

a. Write algorithm for Pass 1 of two pass assembler.

b. Explain advanced assembler directives with examples.

Answers:

a. The algorithm for pass-I of two pass assembler is as follows:

1. LC = 0 (default value)
   littab_ptr = 1;
   pooltab_ptr = 1;
   POOLTAB [1] : first : = 1 : POOLTAB[1] # literals : = 0
2. While the next statement is not an END statement:
   i. If a symbol is present in label field then,
      this_label = symbol in label field:
      Make an entry (this label <LC>, -) in SYMTAB.
   ii. If an LTORG statement then
      a. If POOLTAB [pooltab_ptr] # literals > 0 then process the entire LITTAB [POOLTAB [pooltab_ptr] first].
         LITTAB [littab_ptr_1] to allocate memory to the literal, put address of allocated memory area in the address field of the LITTAB entry and update the address contained in location counter accordingly
      b. pooltab_ptr = pooltab_ptr + 1;
      c. POOLTAB [pooltab_ptr], first : = littab_ptr;

Date

POOLTAB [pooltab_ptr] . # literals := 0;

iii. If a START or ORIGIN statement then
LC := value specified in operand field;

iv. If an EQU statement then,
a. this := value of <address specification>;
b. correct SYMTAB entry for this label to (this.label1, this.addr1)

v. If a declarative stmt then
a. Invoke routine whose id is mentioned in the mnemonic info field. This routine returns code & size.
b. If a symbol is present in label field, correct the symtab entry for this.label1 to (this.label1 <LC> size).
c. LC := LC + size;
d. Generate IC for declaration statement.

vi. If an imperative statement then
a. code := machine opcode from mnemonic info of OPTAB.
b. LC := LC + instruction length from the mnemonic info field of OPTAB.
c. If operand is a literal then
this_literal := literal in operand field:
if POOLTAB : [pooltab_ptr] # literals = 0 or this_literal doesn't match any literal in the range LITTAB [POOLTAB] [pooltab_ptr] first --- LITTAB [littab_ptr] then:
LITTAB [littab_ptr] -- , value := this.literal:
POOLTAB [pooltab_ptr] : # literals + 1 :

Date

litlab_ptr := litlab_ptr + 1
else (i.e operand is a symbol)
this_entry := SYMTAB entry number of operand;
Generate IC for imperative statement

3. Processing of END statement:
a. Perform actions (i) - (iii) of step 2(b)
b. Generate IC for the END statement.

b.

1. ORIGIN (ORG):
- Used to indirectly assign value to symbols.
- When this statement is encountered during assembly of program, the assembler resets its location counter to the specified value.
- Normally, when an ORG without specified value is encountered, the previously saved location counter value is restored.
- syntax : ORIGIN <add spec>
where <add spec> is an <operand spec> or <constant>.
- eg: SYMBOL : 6 byte
VALUE : 1 word
FLAGS : 2 bytes

2. EQU:
- One common use of EQU is to established symbol name that can be used for improved readability in place of numeric values.
- Another use of EQU is in defining mnemonic names for registers.

Date :

- eg: A EQU 0
  X EQU 1
- These statements cause the symbols A, X --- to be entered into SYMBOL with their corresponding values 0, 1.

3. LTORG:
- It allows placing literals into a pool at some other location in the object program.
- Directive LTORG creates literal pool that contains all of the literal operands used since previous LTORG or the beginning of the program.
- Literals placed in a pool by LTORG will not be repeated in the pool at the end of program.
- Assembler allocates memory to the literals of a literal pool. The pool contains all literals used in program since start of program or since the last LTORG statement.

11/23