Arno ir Tado dissasemblerio dokumentacija

Arnas, Tadas

2023 m. gruodžio 3 d.

Turinys

1 Naudojimas **3**

2 Funkcijų aprašai **4**

2.1 read_argument 4
2.2 loop_over_argument 4
2.3 open_input_file 4
2.4 loop_over_bytes 4
2.5 read_bytes 4
2.6 get_byte 4
2.7 read_buffer 4
2.8 write_to_buff 5
2.9 write_to_file 5
2.10 write_to_line 5
2.11 end_line 5
2.12 force_write_to_file 5
2.13 *add_** 5
2.14 reset_double_byte_number 5
2.15 number_to_hex 5
2.16 double_byte_number_to_hex 6
2.17 convert_half_byte_to_HEX 6
2.18 convert_to_decimal 6
2.19 find_word_register 6
2.20 find_byteregister 6
2.21 find_effective_address_registers 6
2.22 find_seg_register 6
2.23 find_poslinkis 6
2.24 find_fill_effective_address 7
2.25 full_reg_detector 7
2.26 full_r_m_detector 7
2.27 CONVERT_dw_mod_reg_r_m_poslinki 7
2.28 CONVERT_w_bojb_bovb 7
2.29 CONVERT_sr 7
2.30 CONVERT_reg 7
2.31 CONVERT_poslinkis 7
2.32 CONVERT_sw_mod_r_m_ poslinkis_bojb_bovb 8
2.33 CONVERT_w_mod_reg_r_m_poslinkis 8
2.34 CONVERT_numeris 8
2.35 check_commands 8

3 Naudojami kintamieji **9**

1 Naudojimas

Programa paleidžiama su 2 argumentais:

1. .com failas kuris turi buti dissassemblinamas
2. .asm failas i kuri reikia irasyti rezultata

Programa skaito *.COM* failus ir juos atkoduoja i 16bitu x86 asemblerio komandas
Failu apribojimai tokie kokie yra DOS operacinei sistemai(failo pavadinimas apribojimas į 8 simbolius ir failu pletinys 3 simboliai atskirti tašku). *.COM* failo apribojimai tokie kokie yra pagal standarta(failas apribojamas į 65280 baitų) Jei pateikiama daugiau nei 2 argumentai like argumentai ignoruojami. Jei pateikamas argumentas */?* tuomet atspauzdinamas trumpas programos aprašas.

2 Funkcijų aprašai

2.1 read_argument

Funkcija į *argument* kintamajį irašo tai, kas buvo pateikta kaip argumentas į komandinę eilutę.

2.2 loop_over_argument

Funkcija iš kintamojo *argument* išskaido į skaitomojo failo pavadinimą ir rašomojo failo pavadinimą *fn_in* ir *fn_out* atitinkamai. Jei argumentų kiekis daugiau nei 2 like argumentai ignoruojami.

2.3 open_input_file

Atidaro failą, su pavadinimu esančiu *fn_in* ir jo handleį išsaugo į *fh_in*.

2.4 loop_over_bytes

Funkcija iškviečia funkciją *read_bytes* ir tada nuolat ciklina funkcijos *check_commands* iškvietimą iki tol, kol pasiekiama failo pabaiga.

2.5 read_bytes

Funkcija nustato ar buvo panaudotas antrasis baitas esantis buferyje. Jei taip *cx* nustatomas į 1, kitu atvėju į 2. Tuomet ciklišKai yra nustatomas *byte_* ir iškviečiama funkcija *get_byte*. Kuomet pasiekiama *fh_in* failo pabaiga, funkcija nutraukia veikimą.

2.6 get_byte

Funkcija nustato ar ar jau pasiekta buferio pabaiga. Jei taip, tuomet *index* yra nunulinamas ir kviečiama funkcija *read_buffer* ir atnaujinama *read_symbols* reikšmė. Į *AL* irašomas sekantis baitas esantis *buff* buferyje. Gauta reikšmė išsaugoma į *next_byte* ir padidinamas *index*. Jei pasiekta failo pabaiga, tuomet *file_end* nustatomas į 1.

2.7 read_buffer

Funkcija nuskaito 200 baitų dydžio bloką iš *fh_in* ir jį išsaugo į *buff*.

2.8 write_to_buff

Funkcija patikrina ar pridėjus teksto eilutę yra daugiau nei 200 simbolių, jei taip, tuomet kviečiama funkcija *write_to_file* ir nunulinamas *write_index*. Jei simbolių kiekis nesiekia 200, tuomet prie bufferio *line* prijungiama eilutė esanti *write_buff* ir nunulinamas *line_length*.

2.9 write_to_file

Funkciją į failą *fh_out* įrašo *write_index* simbolių, kurie yra bufferyje *write_buff*.

2.10 write_to_line

Funkcija naudoja *ptr_* kurią interpretuoja, kaip rodyklę į simbolių masyvą ir visas reikšmes iki NUll *(hex 0)* išsaugo *line*.

2.11 end_line

Funkcija prie *line* prideda *cartridge return* ir *line feed* simbolius. Po to kviečia funkciją *write_to_buff*.

2.12 force_write_to_file

Funkcija iškviečia *end_line* ir *write_to_file* funkcijas. Po to *write_index* yra nunulinamas.

2.13 *add_**

Visus funkcijos pavadinimu *add_** prie *line* prideda tam tikrą simbolį. Pvz: *add_plus* prie *line* prijungia simbolį *+*.

2.14 reset_double_byte_number

Funkcija nunulina kintamąjį *double_byte_number*.

2.15 number_to_hex

Funkcija, iškviesdama funkciją *convert_half_byte_to_HEX* pavercia skaičių esantį *binary_number* į šešioliktainį ir prie *line* prideda raide *h*.

2.16 double_byte_number_to_hex

Funkcija atlieką tą pačią funkciją kaip *number_to_hex*, tik su 2B dydžio kintamuoju.

2.17 convert_half_byte_to_HEX

Funkcija pusbaitį paverčia į šešioliktainį skaičių ir jį parašo ji *line* pabaigoje.

2.18 convert_to_decimal

Funkcija skaičių, esantį *double_byte_number* ir jį irašo į *number_in_ASCII*.

2.19 find_word_register

Funkcija tikrina reikšmę esančią *register_index* ir iš jos gauną registrą. Tuomet atitinkamai į tai koks registras, *ptr_* nustato į pradžią kintamojo, kuriame laikomas jo pavadinimas, kviečiama funkcija *write_to_line*.

2.20 find_byteregister

Funkcijos veikimas analogiškas funkcijai *find_word_register*, tik registrai yra 1B dydžio.

2.21 find_effective_address_registers

Funkcija nustato, kaip adresuojama operaciją *(BX + SI, BX + SI, et allium)* iš *reg_* kintamojo. Nustato *ptr_* į kintamąjį, kuris saugo pavadinimą ir kviečia funkciją *write_to_line*.

2.22 find_seg_register

Funkcija nustato ar yra segmento perrašymas naudojant kintamąjį *register_index*. Nustato *ptr_* į kintamąjį, kuriame laikomas pavadinimas ir kviečiama funkcija *write_to_line*.

2.23 find_poslinkis

Funkcija, naudojant *mod_* nustato kokio dydžio poslinkis, kviečia *read_bytes* atitinkamą kiekį kartų ir kviečia atitinkamą *to_hex* funkciją(*double_byte/number*).

2.24 find_fill_effective_address

Funkcija iškviečia *add_left_bracket*, *find_effective_address_registers*. Jei *mod_* nelygus 0 tuomet papildomai kviečiamos funkcijos: *add_plus*, *find_poslinkis*. Abiejais atvejais funkcija taip pat iškviečia *add_right_bracket*.

2.25 full_reg_detector

Funkcija tikrina ar *w_* lygus 1. Jei taip, tuomet kviečiama funkcija *find_word_register*, kitu atveju kviečiama *find_byte_register*.

2.26 full_r_m_detector

Funkcija tikrina ar *mod_* lygus 3. Jei taip tuomet kviečiama funkcija *full_reg_detector*, kitu atveju kviečiama funkcija *find_full_effective_address*.

2.27 CONVERT_dw_mod_reg_r_m_poslinki

Funkcija dar neimplementuota!

2.28 CONVERT_w_bojb_bovb

Funkcija kviečia *read_bytes* priklausomai nuo *w_* vieną arba du kartus, bitas/-ai paverčiamas į dešimtainį skaičių per funkciją *convert_to_decimal*.

2.29 CONVERT_sr

Funkcija reikšmę esančią *sr_* patalpina į *register_index*, tuomet iškviečia *find_seg_register*.

2.30 CONVERT_reg

Funkcija *reg_* patalpina į *register_index*, tuomet iškviečia *find_word_register*.

2.31 CONVERT_poslinkis

Funkcija nustato *mod_* reikšme į 1 ir kviečia *find_poslinkis*.

2.32 CONVERT_sw_mod_r_m_poslinkis_bojb_bovb

Funkcija dar neimplementuota!

2.33 CONVERT_w_mod_reg_r_m_poslinkis

Funkcija *reg_* reikšmę patalpina į *register_index*, tuomet kviečia funkcijas: *add_space_line*, *full_reg_detector*, *add_comma_line*, *add_space_line*, *full_r_m_detector*.

2.34 CONVERT_numeris

Funkcija *com_num_* patalpina į *double_byte_number* tuomet jį paverčia į dešimtainį skaičių, su *convert_to_decimal*.

2.35 check_commands

Funkcija analizuoja *byte_* ir nustato kokia operacija užkoduota, tuomet nustato jos parametrus*(mod, d, w, et allium)*.

3 Naudojami kintamieji