Uživatelská dokumentace programu

CorW, verze 1.0

Anotace

Program *CorW* je simulátorem souboje několika programů v abstraktním paměťovém poli. Těmto soubojům se někdy říká *Core Wars*. Program je jedním ze zástupců tzv. *MARS* (*Memory Array Redcode Simulator*), které byly inspirovány článkem od A. K. Dewdney v květnovém čísle Scientific American z roku 1984. Jednotlivé programy (dále bojovníci) jsou napsány ve speciálním jazyce *Redcode*, který je podobný zjednodušenému assembleru. Poté co jsou bojovníci nahráni do paměťového pole, simulátor začne střídavě spouštět jejich instrukce. Existuje jedna speciální instrukce, kterou nelze spustit. Bojovníci se snaží tuto instrukci přímo či nepřímo podstrčit soupeři tak, aby ji v příštím cyklu spustil. To by totiž vedlo k jeho zničení a vyřazení z bitvy. Bojovníci se proti zničení brání kopírováním sebe sama na jiná místa v paměťovém poli nebo spouštěním více instancí svého procesu. Existuje mnoho bojových strategií – někteří bojovníci jsou malí a agresivní, jiní jsou složeni z velkého počtu instrukcí a dokážou měnit své chování. Od vydání zmíněného článku v roce 1984 se začaly objevovat snahy pravidla Core Wars ujasnit a standardizovat. První *International Core Wars Standard* (*ICWS*) z roku 1986 byl upraven v roce 1988 (ICWS ’88) a následně v roce 1994 rozšířen dodnes rozpracovaným konceptem, který byl nakonec ustálen na verzi 3.3. Od té doby se jednotlivé simulátory začaly od standardu odchylovat implementací nejrůznějších extenzí. Program CorW v sobě implementuje co nejpřesněji standard ICWS ‘94 3.3.

Úvod do Redcode

Jak již bylo řečeno v anotaci, program je implementací standardu ICWS ‘94 3.3 ve své originální formě bez použití jakýchkoliv extenzí. Objemnost zmíněného standardu neumožňuje jeho doslovný překlad z angličtiny a vložení do této dokumentace. Navíc by tím utrpěla srozumitelnost originálu. Pro porozumění Redcode a této dokumentaci je však nutné předpokládat alespoň zběžnou znalost standardu. Pro rychlé seznámení se základní sémantikou Redcode slouží následující odstavce textu. Pro opravdové ovládnutí Redcode na úrovni programátora bojovníků je však nutné přečíst přímo standard, který sémantiku a syntaxi vysvětluje mnohem detailněji.

Soubor bojovníka se skládá z komentářů a instrukcí. Komentáře mohou být na samostatném řádku nebo následovat za instrukcí. Soubor musí začínat komentářem *;redcode*. Každý bojovník se skládá z jedné a více instrukcí. Každá instrukce je na samostatném řádku. Instrukce je složena z instrukčního kódu, modifikátoru a dvou operandů oddělených čárkou. Instrukční kód a modifikátor jsou odděleny tečkou. Každý operand je složen z adresního režimu a aritmetického výrazu, který může obsahovat závorky, unární minus, sčítání, odčítání, násobení, celočíselné dělení a operaci modulo. Všechny adresní režimy jsou relativní, takže adresa je vzdáleností od současné instrukce. Před instrukcí může být volitelně napsán textový řetězec bez bílých znaků, který při použití v aritmetickém výrazu operandu libovolné instrukce zastupuje číselné pořadí instrukce, před kterou je zapsán ve zdrojovém kódu. Dále se mohou ve zdrojovém kódu nacházet pseudo instrukce a rezervované textové řetězce, jejichž význam bude vysvětlen dále.

Instrukční kódy:

DAT Nelze spustit. Pokus o její spuštění vyřadí jeden bojovníkův proces z bitvy.

MOV Obsah adresy, na kterou ukazuje druhý operand, nahradí obsahem adresy, na kterou ukazuje první operand. Následující instrukce je přidána do fronty.

ADD Obsah adresy, kam ukazuje druhý operand, přepíše součtem obsahu adresy, kam ukazuje druhý operand a obsahu adresy, kam ukazuje první operand. Následující instrukce je přidána do fronty.

SUB Dtto, ale zapisuje se rozdíl.

MUL Dtto, ale zapisuje se součin.

DIV Dtto, ale zapisuje se celočíselný podíl. Dělení nulou má stejný efekt jako pokus o spuštění instrukce DAT.

MOD Dtto, ale zapisuje zbytek po celočíselném dělení. Dělení nulou má stejný efekt jako pokus o spuštění instrukce DAT.

JMP Zařadí do fronty instrukci na adrese, kam ukazuje první operand.

JMZ Otestuje, zda na adrese, kam ukazuje druhý operand, je nula. Pokud ano, tak do fronty zařadí instrukci na adrese, kam ukazuje první operand. Jinak do fronty zařadí následující instrukci.

JMN Dtto, ale testuje nenulovost.

DJN Dtto, ale před posouzením nenulovosti hodnotu dekrementuje.

SEQ Porovná obsah adres, na které ukazují oba operandy. Pokud se hodnoty shodují, přeskočí následující instrukci a do fronty zařadí až jejího následníka. Jinak do fronty zařadí následující instrukci. Z důvodů zpětné kompatibility lze zapsat i jako CMP.

SNE Dtto, ale přeskakuje, když se hodnoty nerovnají.

SLT Dtto, ale přeskakuje, když je první hodnota menší než druhá.

SPL Zařadí do fronty následující instrukci a instrukci, která se nachází ve vzdálenosti odpovídající obsahu adresy, na kterou ukazuje první operand. Přidává bojovníkovy jeden proces.

NOP Zařadí do fronty následující instrukci.

Modifikátory:

A Za obsah adresy, na který je odkazováno operandy, je považován pouze aritmetický výraz prvního operandu instrukce na dané adrese.

B Dtto, ale za obsah je považován druhý operand.

AB Za obsah adresy, na který je odkazováno prvním operandem, je považován pouze aritmetický výraz prvního operandu instrukce na dané adrese. Za obsah adresy, na který je odkazováno druhým operandem, je považován pouze aritmetický výraz druhého operandu instrukce na dané adrese.

BA Za obsah adresy, na který je odkazováno prvním operandem, je považován pouze aritmetický výraz druhého operandu instrukce na dané adrese. Za obsah adresy, na který je odkazováno druhým operandem, je považován pouze aritmetický výraz prvního operandu instrukce na dané adrese.

F Za obsah adresy, na který je odkazováno operandy, je považován aritmetický výraz prvního a druhého operandu instrukce na dané adrese.

X Za obsah adresy, na který je odkazováno prvním operandem, je považován aritmetický výraz prvního a druhého operandu instrukce na dané adrese (v tomto pořadí). Za obsah adresy, na který je odkazováno druhým operandem, je považován aritmetický výraz druhého a prvního operandu instrukce na dané adrese (v tomto pořadí).

I Za obsah adresy, na který je odkazováno operandy, je považována celá instrukce na dané adrese.

Adresní režimy:

\# *Immediate*
Adresa operandu je nulová. Operand tedy ukazuje sám na sebe.

$ *Direct*

Hodnota aritmetického výrazu v operandu je adresa, na které se již nachází cíl. Výchozí režim při neuvedení adresního režimu před operandem.

* *A Indirect*
Výslednou adresou je součet hodnoty aritmetického výrazu operandu a hodnoty aritmetického výrazu prvního operandu instrukce, na kterou tento operand ukazuje.

@ *B Indirect*
Dtto, ale přičítá se hodnota aritmetického výrazu druhého operandu.

{ *A Indirect Predecrement*
Výslednou adresou je součet hodnoty aritmetického výrazu operandu a hodnoty aritmetického výrazu prvního operandu instrukce, na kterou tento operand ukazuje, snížené o jedna.

< *B Indirect Predecrement*
Dtto, ale přičítá se hodnota aritmetického výrazu druhého operandu snížená o jedna.

} *A Indirect Postincrement*
Výslednou adresou je součet hodnoty aritmetického výrazu operandu a hodnoty aritmetického výrazu prvního operandu instrukce, na kterou tento operand ukazuje, která je po provedení součtu zvýšena o jedna.

> *B Indirect Postincrement*
Dtto, ale přičítá se hodnota aritmetického výrazu druhého operandu, která je po provedení součtu zvýšena o jedna.

Pseudo instrukce:

ORG Je následována číslem nebo aritmetickým výrazem, jehož hodnota udává inicializační instrukci bojovníka. Pokud je tato pseudo instrukce vynechána, bojovník je inicializován nultou instrukcí.

EQU Je předcházena textovým řetězcem bez bílých znaků a následována textovým řetězcem, ve kterém již bílé znaky mohou být. Všechny výskyty prvního řetězce ve zdrojovém kódu budou při překladu nahrazeny druhým textovým řetězcem.

END Nepovinná pseudo instrukce označuje konec zdrojového kódu bojovníka. Volitelně může být následována číslem nebo aritmetickým výrazem se stejným významem jako u ORG.

Rezervované textové řetězce:

CORESIZE	Velikost paměťového pole.
MAXCYCLES	Počet cyklů bitvy před vyhlášením remízy.
MAXLENGTH	Maximální počet instrukcí, ze kterých je složen bojovník.
MAXPROCESSES	Maximální počet spuštěných instancí jednoho bojovníka.
MINDISTANCE	Minimální počet instrukcí mezi prvními instrukcemi dvou bojovníků.

Další pojmy:

Initial instruction	Instrukce, kterou je inicializováno paměťové pole. Může být nastavena na RANDOM nebo NONE, což znamená, že pole nebude po předchozích bitvách čištěno.
Separation	Pevný počet instrukcí mezi prvními instrukcemi dvou bojovníků. Navzájem se vylučuje s MINDISTANCE, které umožňuje náhodnou vzdálenost mezi bojovníky.
Read Distance	Maximální vzdálenost, ze které může být prováděno čtení při vyhodnocování operandů instrukce. Pokud adresa ukazuje za tento limit, je ohnuta pomocí operace modulo tak, aby ke čtení došlo v povolené vzdálenosti.
Write Distance	Dtto, ale limituje zápis.

Jako příklad je vhodné uvést jednoduchého bojovníka Dwarf, který byl navržen samotným A. K. Dewdneym v původním článku v Scientific American. Jeho strategií je zapsat nulu do druhého operandu každé čtvrté instrukce v paměťovém poli.

```
;redcode
        ORG     start                   ;Druhá instrukce je inicializační.
step    EQU     4                       ;Všechny výskyty step budou nahrazeny číslem 4.
target  DAT.F   #0,     #0              ;Nic nedělá. Slouží k ukládání cílových pozic.
start   ADD.AB  #step,  target          ;Přičte hodnotu step ke druhému operandu target.
        MOV.AB  #0,     @target         ;Do druhého operandu instrukce, kam ukazuje druhý
                                        ;operand target, uloží nulu.
        JMP.A   start                   ;Skočí zpátky na start.
        END                             ;Ukončení bojovníka.
```

Ovládání

Kliknutím na tlačítko *Add warrior* lze docílit načtení bojovníka do paměťového pole. Po kliknutí na tlačítko dojde k zobrazení dialogového okna pro výběr souboru. Lze vybrat pouze soubory s příponou *.red*, což je tradiční přípona souborů obsahující Redcode instrukce. Po potvrzení výběru souboru dojde k pokusu o jeho překlad. Je kontrolováno, zda soubor začíná řetězcem *;redcode*, který potvrzuje, že se jedná o Redcode soubor. Pokud je v souboru chyba, neodpovídá standardu nebo vůbec neobsahuje validní instrukce Redcode, je zobrazena chybová hláška. Pokud vše proběhlo v pořádku, bojovník se objeví na mapě paměťového pole a do seznamu bojovníků v pravé části okna programu jsou přidány jeho ovládací prvky. Jedná se především o tlačítko, jehož barva je stejná jako barva paměťových buněk, které byly bojovníkem ovlivněny. Popisek tlačítka odpovídá pořadí, v jakém byl bojovník přidán do pole. Po stisknutí tlačítka se zobrazí zdrojový kód a přeložené instrukce daného bojovníka. Dále má každý bojovník vedle tlačítka zobrazeno naplnění jeho úlohové fronty. Před spuštěním simulace má každý bojovník ve frontě jednu úlohu.

Simulaci lze spustit, jakmile jsou do paměťového pole nahráni alespoň dva bojovníci. Ke spuštění simulace dojde buď po stisknutí tlačítka *Start*, nebo tlačítka *Step.* Po stisknutí tlačítka Start bude simulace probíhat tak dlouho, dokud neskončí bitva. Tlačítko Step provede pouze jeden cyklus bitvy, tedy spuštění jedné instrukce od každého bojovníka. Bitva může být kdykoliv pozastavena tlačítkem *Stop* a následně opět spuštěna. Rychlost bitvy lze regulovat posuvníkem nad tlačítky – čím více je nastaven doprava, tím rychleji bude bitva probíhat. Pod tlačítky je zobrazen počet cyklů, které již byly spuštěny, a počet cyklů, které zbývají do konce bitvy. Bitva však může skončit i dříve, pokud zůstane pouze jeden živý bojovník.

Mapa paměťového pole je pokryta malými čtverci, které odpovídají jednotlivým buňkám paměťového pole. Čtverce mění svoji velikost podle toho, jak rozsáhlé je paměťové pole. Černý čtverec reprezentuje prázdnou buňku paměti, tedy takovou, která doposud nebyla ovlivněna žádným bojovníkem. Obarvené buňky odpovídají svojí barvou bojovníkovi, který je naposledy ovlivnil. Existují dvě rezervované barvy, které nejsou používány pro bojovníky. První z nich je červená, která ukazuje pro každého bojovníka instrukci, která bude v příštím cyklu spuštěna. Druhou barvou je modrá, která po najetí myši na bojovníkovo barevné tlačítko obarví jeho příští instrukci. Protože v dané chvíli budou příští instrukce ostatních bojovníků červené, lze takto nalézt konkrétní místo, kde se daný bojovník zrovna nachází.

V pravé horní části okna programu se nachází podrobný náhled na obsah paměťového pole na zvolené adrese. Adresa lze zvolit buď zadáním čísla do políčka nad náhledem, nebo kliknutím myši do mapy paměťového pole. V černém rámečku je vždy zobrazeno 10 instrukcí od této adresy. Barvy instrukcí odpovídají obarvení buněk v mapě paměťového pole. Pro rychlejší orientaci lze aktivovat pomocný *Slider*, který červeně obarví buňky, které jsou zrovna zobrazeny v náhledu. Slider je viditelný pouze pokud neběží simulace.

Parametry simulace lze přenastavit po kliknutí na tlačítko *Setup*. V dialogu, který se zobrazí, je možné nastavit hodnoty buď manuálně, anebo kliknutím na jeden ze tří výchozích profilů načíst nejpoužívanější konfigurace. Význam jednotlivých položek je vysvětlen ve standardu. Po potvrzení

dialogu tlačítkem *OK* jsou hodnoty zkontrolovány a načteny. Pokud je některá z hodnot odmítnuta, objeví se informační hláška. Po stisknutí tlačítka OK dojde k resetování aplikace podle nového nastavení – dojde k vyčištění paměťového pole a seznamu bojovníků.

Pro resetování aplikace po bitvě lze použít také tlačítko *Wipe*, které resetuje aplikaci podle současného nastavení hodnot v Setup menu.