Mel, egy Igazi Programozó története[1]

Ezt először 1983. május 21-én töltötték fel a Usenetre[2]

Nemrég megjelent egy cikk a programozás macsó vonásairól,
benne egy egyszerű és nyers kijelentéssel, miszerint:

Az Igazi Programozók FORTRAN-ban kódolnak.[3]

Lehet, hogy most igen,
a diétás sörök, zsebszámológépek és „felhasználóbarát" szoftverek
hanyatló korában,
de a Régi Szép Időkben,
amikor a „szoftver" szó még furcsán csengett,
és az Igazi Számítógépeket mágnesdobokból[4] és elektroncsövekből építették,
akkoriban az Igazi Programozók gépi kódban programoztak.
Nem FORTRAN-ban. Nem RATFOR-ban. Még csak nem is assembly nyelven.
Gépi Kódban.
Nyers, puritán, rejtélyes hexadecimális számokkal.
Közvetlenül.

Nem lenne jó, ha programozóknak egy új generációja
nőne fel e dicső múlt ismerete nélkül,
ezért kötelességemnek érzem, hogy elmondjam,
amennyire a generációs szakadékot átívelve csak tudom,
hogyan kódolt egy Igazi Programozó.
Melnek fogom hívni,
mert ez volt a neve.[5]

---

[1] Az eredetileg névtelenül megjelent „The Story of Mel, A Real Programmer" című írás szerzője Ed Nather: https://en.wikipedia.org/wiki/Ed_Nather
Az írást Erik Brunvand ellátta magyarázatokkal és közzétette itt: https://www.cs.utah.edu/~elb/folklore/mel-annotated/mel-annotated.html
Részben ez utóbbit felhasználva készítette a magyar fordítást Németh Krisztián, mely letölthető a http://mesz-i.org Publikációk oldaláról. A fordítás átnézését köszönöm Szigethy Katalinnal és Képes Gábornak.

[2] A Usenet egy nagyon korai számítógépes üzenőfalrendszer volt. Az írás 1983-as, amikorra a programozás negyedszázaddal korábbi hőskora már kezdett feledésbe merülni.

[3] A szerző itt Ed Post : Real Programmers Don't Use Pascal c. tréfás cikkére utal, amely 1983-ban jelent meg a Datamation Magazine-ban és az Interneten ma is sok helyen fellelhető. A cikkben a Fortran képviselte a „régi szép időket", és a Pascal a „hanyatló modern kort". Ma már mindkettő réginek számít, azonban ezen írás szerzője szerint már a Fortran is „hanyatlóan modern" volt…

[4] A mágnesdob egy forgó fémhenger volt, melynek az oldalára szerelt fejek rögzítették a felületére az információt. Felépítésében inkább a későbbi merevlemezekre hasonlított – bár nem egy mozgó, hanem sok rögzített író-olvasó feje volt –, azonban a kezdeti számítógépekben jobb híján ezt használták operatív memóriának. Keservesen lassú volt.

[5] 1999-ben derült ki, hogy Mel teljes neve Melvin Kaye. Ld. http://catb.org/jargon/html/story-of-mel.html és https://en.wikipedia.org/wiki/The_Story_of_Mel

Akkor találkoztam először Mellel,
amikor a Royal McBee Computer vállalathoz[6] mentem dolgozni,
amely az írógépgyártó cég mára megszűnt leányvállalata volt.
Ez a vállalat gyártotta az LGP-30-at,
egy (akkori viszonylatban) kicsi, olcsó
mágnesdob-memóriás számítógépet,
és akkor kezdte el gyártani
az RPC-4000-et, egy sokkal fejlettebb,
nagyobb, jobb, gyorsabb – mágnesdob-memóriás számítógépet.
A ferritgyűrűs memória[7] túl sokba került,
és amúgy is csak átmeneti megoldást jelentett.[8]
(Nos, ezért nem hallottál sem a cégről,
sem a számítógépről.)

Az én feladatom egy FORTRAN fordító megírása volt
erre a gyönyörűségre, és Mel volt az, aki megismertetett a csodáival.
Mel nem szerette a fordítókat.

– Ha egy program nem tudja átírni a saját kódját – kérdezte –
akkor mi értelme van?[9]

Mel írta,
hexadecimálisban[10],
a cég legnépszerűbb számítógépprogramját,
ami az LGP-30-on futott
és blackjacket játszott a vevőjelöltekkel
a számítástechnikai kiállításokon.
A hatása mindig drámai volt.
Az LGP-30 stand minden kiállításon tömve volt,
miközben az IBM-es eladók csak ácsorogtak
egymással cseverészve.
Az a kérdés, hogy ez vajon segített-e valaha akár csak egy gépet is eladni,
soha fel sem merült.

Mel feladata volt átírni
a blackjack programot az RPC-4000-re.
(Hordozhatóság? Az mi fán terem?)
Az új számítógépnek „egy-plusz-egy-címes”
címzési módja volt,
amelyben minden gépi utasítás

---

[6] https://en.wikipedia.org/wiki/Royal_Typewriter_Company#Computers

[7] A ferrittár jelentette a következő lépést a számítástechnika fejlődésében. A kis gyűrűkből kézzel szőtt memóriamodulok körülbelül ötven-százszor gyorsabbak voltak a mágnesdobos elődeiknél.

[8] Ezt mondták a cég mérnökei, és ez nagyon rossz döntésnek bizonyult.

[9] Önmódosító kódot ma már nem szokás írni, de régen ez másképp volt.

[10] A gépi utasításkódokat és az adatokat közvetlenül, tizenhatos számrendszerben leírva.

az utasításkódon és az opernadus címén kívül
tartalmazott egy második címet is, amely jelezte, hogy a forgódobon hol
helyezkedik el a következő utasítás.

Modern szavakkal kifejezve,
mindegy egyes utasítást követett egy GO TO!
Na erre varrjon gombot a Pascal!

Mel imádta az RPC-4000-et,
mert optimalizálni tudta a kódját,
azaz úgy tudta elhelyezni az utasításokat a dobon,
hogy amikor az egyik futtatása befejeződött,
a következő épp az „olvasófej” alá ért,
és így azonnal végre is lehetett hajtani.
Volt is egy program e célra,
egy „optimalizáló assembler”,
de ezt Mel nem volt hajlandó használni.

– Sosem tudhatod, hova fogja tenni a dolgokat – magyarázta –
ezért külön konstansokat kellene használnod.

Sok idő telt el, mire megértettem ezt a megjegyzést.
Mivel Mel az összes utasításkód
numerikus értékét ismerte,
és maga osztotta ki a mágnesdobon a címeket,
ezért minden utasítást, amit leírt,
tekinthetett számkonstansnak is.
Foghatott mondjuk egy korábbi „összeadás” utasítást,
és szorozhatott vele,
ha a numerikus értéke a megfelelő volt.[11]
Az ő kódját más nem tudta egykönnyen módosítani.

Összehasonlítottam Mel kézzel optimalizált programjait
azzal a kóddal, amit az optimalizáló assembler gyúrt össze,
és Melé mindig gyorsabb volt.
Azért volt ez, mert a programtervezés „top-down” módszerét
még nem találták ki,
és Mel amúgy sem használta volna.
Ő a programja ciklusainak legbelső részét írta meg először,
hogy először azokat lehessen
optimális helyekre tenni a mágnesdobon.
Az optimalizáló assembler nem volt elég okos, hogy így csinálja.

[11] A gépi utasításokat is számkonstansokként tárolták a memóriában, és ez ma is így van. Ma ezeket az adatoktól elkülönítve tartjuk, akkor azonban nem voltak ilyen megkötések. Ugyanakkor a memória szűkös és a végrehajtás lassú volt, így az efféle optimalizálás jelentősége nagy volt.

Mel sosem írt késleltető ciklusokat sem,
még akkor sem, amikor a csökönyös Flexowriternek
a helyes működéshez szüksége volt egy kis szünetre a kimeneti karakterek között.[12]
Ő inkább úgy rendezte el az utasításokat a dobon,
hogy a soron következő már épp elhaladt az olvasó fej alatt,
amikor szükség volt rá,
így a dobnak egy újabb teljes fordulatot kellett megtennie
a következő utasítás megtalálásához.
Ki is talált egy feledhetetlen kifejezést erre az eljárásra.
Habár az „optimum” abszolút kifejezés,
akárcsak az „egyedi”, gyakori szófordulat lett
relatív formában:
„nem igazán optimum” vagy „kevésbé optimum”
vagy „nem nagyon optimum”.
Mel a maximális késleltetésű elhelyezkedést
„legpesszimumabb”-nak nevezte.

Miután befejezte a blackjack programot
és működésre bírta
(– Még az inicializáló részt is optimalizáltam[13] –
mondta büszkén),
az értékesítési osztálytól egy változtatási igény érkezett.
A program egy elegáns (optimalizált)
véletlenszám-generátort használt
a „kártyák” keveréséhez és a „paklból” való osztáshoz,
ami néhány eladó szerint túl igazságos volt,
mivel néha a vevők veszítettek.
Azt akarták, hogy Mel módosítsa a programot
úgy, hogy a konzolon egy előlapi kapcsoló átállításával
megváltoztathassák az esélyeket és hagyják a vevőt nyerni.

Mel erre nem volt hajlandó.
Úgy érezte, ez nyilvánvalóan tisztességtelen,
ami igaz is volt,
és összeférhetetlen programozói becsületével,
ami igaz is volt,
ezért nem teljesítette a kérést.
A Vezető Értékesítő beszélt Mellel,
akárcsak a Nagyfőnök, és a főnök noszogatására
néhány Programozó Kolléga.

---

[12] A Friden cég Felxowriter-e (https://en.wikipedia.org/wiki/Friden_Flexowriter) egy egyszerű nyomtató volt, vagyis inkább egy távvezérelhető írógép. Nem rendelkezett memóriával, betűnként kellett elküldeni neki a nyomtatandó szöveget, ráadásul nem volt szabad túl gyorsan, hiszen a mechanikának készen kellett állni egy újabb leütésre az előző után. A szóban forgó számítógép perifériakezelése sem volt túl fejlett, a programozónak kellett biztosítani – szükség szerint késleltető ciklus beiktatásával –, hogy kellően lassan küldje az adatokat a nyomtatóra.

[13] Ez minden bizonnyal már akkor is felesleges volt.

Mel végül ráállt és megírta a kódot,
de a módosítást fordítva készítette el,
így amikor a kapcsoló fel volt kapcsolva,
a program csalt és minden alkalommal győzött.
Mel elégedett volt:
azt állította, hogy a tudatalattija irányíthatatlanul etikus
és mereven elzárkózott attól, hogy kijavítsa a kódot.

Miután Mel távozott a cégtől dú$abb Łegelőket keresve,
a Nagyfőnök engem kért meg, hogy nézzem át a kódot,
hogy megtalálom-e a csaló részt és meg tudom-e fordítani.
Nem túl lelkesen, de megígértem, hogy megnézem.
Mel kódjának végigkövetése igazi kaland volt.

Sokszor éreztem úgy, hogy a programozás egyfajta művészet,
aminek az igazi értéke csak annak tárul fel,
aki szintén járatos e misztikus művészetben;
a folyamat jellegénél fogva
csodálatos gyöngyszemek és briliáns trükkök
maradnak rejtve az emberi szem és csodálat elől, néha mindörökre.
Sokat megtudhatsz egy személyről
pusztán a kódjának átolvasásával,
még hexadecimálisban is.
Mel szerintem egy fel nem ismert zseni volt.

Talán a legmegdöbbentőbb az volt,
amikor találtam egy ártatlan ciklust, amelyben nem volt feltételvizsgálat.
Nulla feltételvizsgálat. Semmi.
A józan ész szerint ennek egy végtelen ciklusnak kellett volna lennie,
amelyben a program örökké, vég nélkül körben jár.
A vezérlés azonban végigment rajta
és simán ki is jutott belőle.
Két hétig tartott, mire rájöttem.

Az RPC-4000 számítógépnek volt egy igazán modern funkciója,
amit index regiszternek hívtak.
Ez lehetővé tette, hogy a programozó olyan ciklus írjon,
amely egy indexelt utasítást tartalmazott:
minden körben
az index regiszterben lévő szám
hozzáadódott az adott utasítás címéhez,
így az a sorban következő adatra hivatkozott.
Csak meg kellett növelni az index regisztert

minden körben.[14]
Mel ezt sohasem használta.

Ehelyett beolvasta az utasítást egy gépi regiszterbe,
hozzáadott egyet a címéhez,
és visszaírta.[15]
Ezután a módosított utasítást közvetlenül a regiszterből
hajtotta végre.
A ciklust úgy írta meg, hogy ezt a többlet végrehajtási időt
is figyelembe vette –
amikor ez az utasítás befejeződött,
a következő éppen a dob olvasófeje alatt volt,
futásra készen.
De a ciklusban nem volt feltételvizsgálat.

Akkor bukkantam a kulcsfontosságú nyomra, amikor észtevettem, hogy
az indexregiszterbitje,
az a bit, amely a cím és az utasításkód között
helyezkedett el az utasításszóban,
be volt billentve –
pedig Mel sosem használta az indexregisztert,
azt[16] mindig nullában hagyta.
A felgyúló fény majdnem megvakított.

Az adatokat, amelyekkel dolgozott, a memóriatartomány
teteje környékére helyezte –
a legmagasabb helyre, amit még az utasítások meg tudtak címezni –
így miután az utolsó adat is feldolgozásra került,
az utasítás címének növelése
annak túlcsordulását eredményezte.
Az átvitel hozzáadott egyet
az utasításkódhoz[17], átírva azt az utasításkészlet következő elemére:
egy ugró utasításra.[18]
Persze a következő programutasítás a
nullás címen volt,
és a program vígan futott tovább.

---

[14] Így lehetett például egy tömb minden elemén hatékonyan végigmenni.
[15] Az utasításhoz tartozó címmezőt módosította és írta vissza az egészet az utasítás eredeti helyére.
[16] Valószínűleg nem az indexregiszterre, hanem az utasításokban lévő indexregiszterbitre utal.
[17] Az utasításszó minden bizonnyal így nézett ki: [utasításkód][indexregiszterbit][cím][következő utasítás címe]. Mivel a címet minden végrehajtás után megnövelte eggyel Mel kódja, az az utolsó adat után túlcsordult, mert ezt az adatot a legmagasabb címezhető memóriaterületen tartotta. Az indexregiszter értéke is egy volt, ezért a túlcsordulás „tovább terjedt”, eggyel megnövelve az utasításkódot és nullába állítva az indexregiszterbitet és a címet is.
[18] Az persze csak szerencse, hogy épp az ugró utasítás kódja volt ez, de Mel ezt tudta és kihasználta.

Nem tartottam a kapcsolatot Mellel,
így nem tudom, beadta-e valaha a derekát a változásoknak, amelyek elemi
erővel söpörtek végig a programozási technikákon,
azok óta a rég elmúlt idők óta.
Szívesen gondolom azt, hogy nem.
Akárhogy is, annyira le voltam nyűgözve, hogy abbahagytam
annak a rakoncátlan módosításnak a keresését,
és megmondtam a Nagyfőnöknek, hogy nem találtam meg.
Nem tűnt meglepettnek.

Amikor otthagytam a céget,
a blackjack program még mindig csalt,
ha átállítottad a megfelelő kapcsolót,
és azt hiszem, ez így van rendjén.
Nem éreztem helyénvalónak
belepiszkálni egy Igazi Programozó kódjába.