Sed kion pri la estonteco? Kompreneble nia unua tasko estas rekolekti la membrojn, reenfluigi la kotizojn kaj tiel certigi la regulan aperadon de SCIENCA REVUO. La estraro ankoraŭ diskutas, ĉu transsalti la jarkolektojn por 1979 kaj 1980 kaj kompense liveri al la membroj iujn aliajn publikigaĵojn kun rilato al scienco, aŭ ĉu aperigi iom reduktitajn jarkolektojn por tiuj du jaroj. Plie ekzistas planoj pri pretigo de varbiloj kaj de membrolisto aŭ jarlibro. La estraro ankaŭ diskutas pri okazigo de sciencistaj simpozioj kaj pri aperigo de sciencaj lernolibroj aŭ aliaj sciencaj verkoj en Esperanto.

Kiel vi vidas, taskoj abundas. Mi invitas ĉiujn esperantistajn sciencistojn al efika kunlaboro por realigi tiujn kaj aliajn planojn.

Kolege salutas vin

C. Støp-Bowitz
prezidanto de ISAE

Notoj de la ĉefredaktoro

Referencante al SR, 1978, 1, 8, mi denove petas, ke ĉiu aŭtoro, kiu deziros aperigi sian traktaĵon en SR, bonvolu atenti jenon:

1. Verki laŭeble pri sia propra originala esplorado tiel, ke ĝin komprenu eĉ alifakaj sciencistoj;
2. Sendi al mi la tekston en 2 ekzempleroj, tajpitaj sur maldika papero;
3. Maŝinskribi normallitere en la tajp-areo de 165 mm × 250 mm, kun la liniado 1½, t.e. kun 40 linioj surpaĝe;
4. Ĉiun vorton neesperantan substreki ondlinie (por la kursiva komposto), dum rektlinia substreko signifas komposton dikliteran;
5. En la teksto noti lokojn, kie troviĝu eventualaj figuroj;
6. Ne forgesi referenc-liston;
7. Aldoni koncizan resumon en sia gepatra lingvo.

Ĉar SR strebas al perfekta faklingvo, mi supozas, ke la verkontoj zorge trastudos koncernan materialon, ĝis nun aperintan en SR, kaj ke ili kunlaboros kun TC-ISAE.

Anticipe mi dankas pro pacienca kunlaboro.

Sincere via

J. Kavka

INTEL 8080 — priskribo kaj programado

Vortoj de la tradukinto

La sekvantaj paĝoj estas grandparte traduko de du ĉapitroj el la usona libreto *"Intel 8080 Microcomputer Systems User's Manual"*, eldonita en septembro 1975, pri kiu ne ekzistas kopirajtoj. La paĝoj konsistigas la ĉefan parton de enkonduko al la kompreno de funkciado kaj utiligo de la fama konata mikroprilaborilo *INTEL 8080*.

Mi laboras ankaŭ pri konekto al komputoraj retoj per la proceduro *X25*, precipe pri la franca, pakkomuta reto *TRANSPAC*.

Ĉiuj kritikoj kaj rimarkigoj estas bonvenaj. Bonvolu sendi ĉiujn poŝtaĵojn al la adreso de mia oficejo:

Christian Bertin, inĝeniero
C.C.E.T.T., departemento *RSI*
2, rue de la Mabilais, B.P. 1266
35013 *Rennes Cedex*, Francio

1. La funkcioj de komputoro

Tiu ĉapitro prezentas iujn bazajn komputajn konceptojn. Ĝi havigas foninformojn kaj difinojn, kiuj estos utilaj en la postaj ĉapitroj de la manlibro. Tiuj, kiuj jam familiariĝis pri komputoroj, povas preterlasi ĝin.

Tipa karaktra komputoro konsistas el:
— centra prilabora unuo (CPU),
— storo,
— enig/elig- (EN/EL) pordoj.

La storon oni utiligas por stori **instrukciojn**, koditajn partojn de informo, kiuj regas la aktivojn de CPU, kaj **datumojn**, la koditajn partojn de informo, kiuj estas prilaborataj de la CPU. **Programo** estas grupo da logike ligitaj instrukcioj, storita en la storo. CPU legas ĉiun instrukcion el la storo laŭ logike determinita sekvenco kaj uzas ĝin por startigi prilaborajn agojn. Se la programsekvenco estas kohera kaj logika, la efektivigo de la programo produktos kompreneblajn kaj utilajn rezultojn.

La storon oni utiligas ankaŭ por stori la datumojn manipulotajn same kiel la instrukciojn, kiuj regos tiun manipuladon. Oni devas organizi la programon tiel, ke CPU ne legu neinstrukcian vorton, kiam ĝi atendas instrukcion. CPU kapablas rapide atingi iun ajn datumon, storitan en la storo; sed ofte la storo ne estas sufiĉe vasta, por ke oni povu stori la tutan datumaron necesan por aparta laboro. Oni povas solvi la problemon provizante la komputoron per unu aŭ pliaj **enigpordoj**. CPU kapablas adresi tiujn pordojn kaj enirigi la tiean datumon. La aldono de enigpordoj ebligas al la komputoro ricevi informojn el eksteraj ekipaĵoj (trubendlegilo aŭ fleksebla disko) laŭ rapidoj kaj kvantoj.

Komputoro bezonas ankaŭ unu aŭ pliajn **eligpordojn**, kiuj ebligas al CPU transdoni la rezultojn de sia prilaboro al la ekstera mondo. La eligaĵo povas iri al vidigilo por uzo fare de homa funkciigisto, al periferia aparato, kiu produktas "firman kopiaĵon", ekz. al linipresilo, al periferia storaparato, kia estas supladiska ilo, aŭ la eligaĵo povas konsistigi prilaboro-regajn signalojn, kiuj regas la funkciadon de alia sistemo, ekz. de aŭtomata muntlinio. Same kiel enigpordojn, ankaŭ eligpordojn oni povas adresi. La kunaj enig- kaj eligpordoj ebligas komunikadon de la prilaborilo kun la ekstera mondo.

CPU unuigas la sistemon. Ĝi regas la funkciojn, efektivigatajn de la aliaj komponaĵoj. CPU devas kapabli legi instrukciojn el la storo, analizi iliajn duumajn enhavojn kaj efektivigi ilin. Ĝi devas kapabli ankaŭ adresi storon kaj enig/eligpordojn, kiam tio estas necesa por la efektivigo de la instrukcioj. Krome, CPU devus kapabli identigi kaj respondi iujn eksterajn regsignalojn, kiaj estas INTERROMP- kaj ATEND-petoj. La funkciaj unuoj en CPU, kiuj ebligas al ĝi efektivigi la funkciojn, estas priskribitaj ĉi-sube.

Tipa centra prilabora unuo (CPU) konsistas el la interligitaj funkciaj unuoj:

— registrumoj,
— aritmetika-logika unuo (ALU),
— regcikvitaro.

Registrumoj estas portempe storaj unuoj en CPU. Iuj registrumoj, kiaj estas la instrukcimontrilo kaj la instrukciregistrumo, havas specialan utilon. Aliaj registrumoj, ekz. la akumulatoro, estas por uzo multe pli ĝenerala.

1.1. Akumulatoro

La akumulatoro kutime enhavas unu el la operandoj manipulotaj de

ALU. Unu tipa instrukcio ordonas al ALU adicii enhavon de iu alia registrumo kun enhavo de la akumulatoro kaj stori la rezulton en la akumulatoron mem. Ĝenerale la akumulatoro estas kaj la fontregistrumo (operando) kaj la celregistrumo (rezulto).

CPU ofte inkludas nombron da pliaj ĝeneralcelaj registrumoj, kiuj povas utili por stori operandojn aŭ provizoran datumon. La havo de ĝeneralcelaj registrumoj forigas la neceson dudirekte movi provizorajn rezultojn inter la storo kaj la akumulatoro, tiel pligrandigante la prilaborajn rapidon kaj efikon.

1.2. Instrukcimontrilo (saltoj, subrutinoj kaj la stako)

La instrukciojn, kiuj konsistigas programon, oni storas en la sistemstoron. La centra prilaborilo adresas enhavon de la storo por determini la taŭgan agon. Tio signifas, ke la prilaboro devas koni la storeron, kiu enhavas la postan instrukcion.

Ĉiuj storeroj estas numeritaj por distingi unujn de la aliaj. La numero, kiu identigas storeron, estas nomata ĝia **adreso**.

La prilaborilo prizorgas registrumon, kiu enhavas la adreson de la posta programinstrukcio. Tiu registrumo estas nomata **instrukcimontrilo**. La prilaborilo aktualigas la instukcimontrilon aldonante "1" al la registrumo ĉiufoje, kiam ĝi legas instrukcion. Tiel la instrukcimontrilo ĉiam estas aktuala (indikanta postan instrukcion).

La programisto do storas siajn instrukciojn en numere apudajn adresojn tiel, ke la malplej altaj adresoj enhavas la unuajn instrukciojn efektivigotajn, kaj la plej altaj adresoj enhavas la plej postajn. La programisto rajtas malobei tiun paŝpostpaŝan regulon nur tiam, kiam instrukcio en storero estas **saltinstrukcio** al alia storero.

Saltinstrukcio enhavas adreson de instrukcio, kiu estas efektivigota tuj post tiu saltinstrukcio. La tuj postan instrukcion oni rajtas stori en iun ajn storeron, la programita salto indikas ĝian ekzaktan adreson. Dum la efektivigo de la saltinstrukcio la prilaborilo anstataŭigos la enhavon de sia instrukcimontrilo per la adreso indikita de la salto. Tiel la logika daŭro de la programo konserviĝas.

Speciala speco de programsalto okazas, kiam la storita programo vokas subrutinon. En tiu speco de salto oni postulas de la prilaborilo memori enha-

von de la instrukcimontrilo en la momento de la salt-okazo. Tio ebligas al la prilaborilo daŭrigi la efektivigon de la ĉefa programo, kiam ĝi ĝisfine efektivigas la lastan instrukcion de la subrutino.

Subrutino estas programo en programo. Kutime ĝi estas ĝeneralcele instrukcisekvenco, kiu povas esti ripete efektivigata dum la ĉefa programo. Rutinaj programoj, kiuj kalkulas la potencon je du, la sinuson aŭ la logaritmon de programvariablo, estas bonaj ekzemploj de funkcioj ofte skribitaj kiel subrutinoj. Aliaj ekzemploj povas esti programoj konceptitaj por enigi aŭ eligi datumojn de aŭ al apartaj periferiaj aparatoj.

La prilaborilo, laŭ speciala maniero, administras subrutinojn por certigi ordan reiron al la ĉefa programo. Kiam la prilaborilo legis vokinstrukcion, ĝi krementas la instrukcimontrilon (kiel kutime), kaj storas ties enhavon en rezervitan stor-areon nomatan **stako**. La stako tiel savas la adreson de la instrukcio, efektivigota post kompleta efektivigo de la subrutino. Poste la prilaborilo movas la adreson, indikitan en la voko, en sian instrukcimontrilon. La poste legota instrukcio do estos la unua paŝo de la subrutino.

La lasta instrukcio en ĉiuj subrutinoj estas **reiro**. Tia instrukcio ne bezonas indiki adreson. Kiam prilaborilo legas reir-instrukcion, ĝi nur anstataŭigas la nunan enhavon de la instrukcimontrilo per adreso, storita ĉe la stakosupro. Tio ebligas al la prilaborilo, daŭrigi la efektivigon de la vokinta programo ĉe la punkto, kiu tuj sekvas la originan vokinstrukcion.

Subrutinoj ofte estas **ŝtuparaj**; tio signifas, ke subrutino kelkfoje vokas alian, duan subrutinon. Tiu dua povas voki trian kaj tiel plu. Tio estas vere akceptebla, se la prilaborilo posedas sufiĉan kapaciton por stori la necesajn reir-adresojn kaj proviĝis per logiko por tion fari. Alidirate, la maksimuma profundo de la ŝtuparo determiniĝas per la profundo de la stako mem. Se la stako disponas pri loko por stori tri reir-adresojn, do tri niveloj da subrutinoj estas akcepteblaj.

Prilaboriloj diversmaniere administras stakojn. Iuj posedas eblon por storadi reir-adresojn interne de la prilaborilo mem. Aliaj prilaboriloj utiligas rezervitan areon de ekstera storo kiel stakon kaj simple prizorgas **montrilan registrumon**, kiu enhavas la adreson de la plej freŝdata staken-movo. La ekstera stato principe ebligas senliman subrutinŝtuparon. Plie se la prilaborilo disponas pri instrukcioj, kiuj ebligas puŝi aŭ tiri la enhavon de la akumulatoro, aŭ de aliaj ĝeneralcelaj registrumoj, en la stakon aŭ el la stako laŭ la adreso, storita en la stakmontrilo, tiam la prilaboro kun plurnivelaj interrompoj (priskribota en ĉi tiu ĉapitro) estas ebla. La prilaborilan staton (t.s. la enhavojn de ĉiuj registrumoj) oni povas savi en la stakon, kiam interrompo akceptiĝas, kaj poste restarigi ĝin, kiam la interrompo ĝisfine serviĝis. La eblo savi la prilaborilan staton en ĉiu momento estas disponebla, eĉ se la interrompserva programo mem estas interrompata.

1.3. Instrukciregistrumo kaj malkodilo

Ĉiu komputoro havas **vortlongon**, kiu karakterizas tiun maŝinon. Komputoro- vortlongon oni kutime determinas per la amplekso de ĝiaj internaj storeroj kaj interkonektaj kanaloj (nomataj **buso**). Ekzemple komputoro, kies registrumoj kaj busoj kapablas stori kaj transmovi ok bitojn da informo, havas karakterizan vortlongon de ok bitoj kaj nomiĝas okbita paralela prilaborilo. La okbita paralela prilaborilo ĝenerale plej efikas, utiligante okbitajn duumajn kampojn, kaj la storo asociita al tia prilaborilo estas organizita por stori ok bitojn en ĉiun adreseblan storeron. Datumoj kaj instrukcioj estas storataj kiel okbitaj duumaj nombroj aŭ kiel nombroj, kiuj estas entjeraj obloj de ok bitoj: 16 bitoj, 24 bitoj k.t.p. Tiu karakteriza okbita kampo estas ofte nomata okbito.

Ĉiu operacio, kiun prilaborilo kapablas efektivigi, estas identigita kiel unika okbito de datumo, konata kiel **instrukcikodo** aŭ **operacikodo**. Okbita vorto, uzata kiel instrukcikodo, permesas distingon inter 256 eblaj agoj, kio estas pli ol sufiĉa por la plimulto de la prilaboriloj.

La prilaborilo legas instrukcion laŭ du apartaj operacioj. Unue la prilaborilo transmovas la adreson el sia instrukcimontrilo al la storo, due la storo resendas la adresitan okbiton al la prilaborilo. CPU storas tiun instrukciokbiton en registrumon, nomatan la **instrukciregistrumo** kaj uzas ĝin por regi la agojn dum la resto de la instrukcia efektivigo.

La meĥanismo, laŭ kiu la prilaborilo tradukas instrukcikodon en specifajn agojn, postulas pli detalan priskribon, ol permesas la ĉi tie disponebla loko. Tamen la koncepto devus esti intuicie klara al ĉiu logikplanisto. La okbitojn, storitajn en la instrukciregistrumon, oni povas malkodi kaj utiligi por selekte aktivigi unu el pluraj eliglineoj, en tiu okazo el ĝis 256 lineoj. Ĉiu lineo prezentas aron de agoj, asociitaj al efektivigo de unu aparta instrukcikodo. La aktivan lineon oni povas kombini kun selektitaj tempantaj pulsoj por estigi elektrajn signalojn, kiuj do povas utili por iniciati specifajn agojn. La tradukon de kodo en agon realigas la instrukci-malkodilo kaj la asociita regcirkvitaro.

La okbita instrukcikodo ofte sufiĉas por specifi apartan prilaboran agon. Tamen kelkfoje la efektivigo de instrukcio postulas pli da informo, ol okbitoj povas enteni.

Ekzemplo de tio estas instrukcio, kiu adresas storeron. La baza instrukcikodo identigas la efektivigotan operacion, sed ne kapablas specifi ankaŭ la celadreson. En tia okazo estas necesa du- aŭ tri-okbita instrukcio. Sinsekvaj instrukci-okbitoj estas storitaj en sinsekvajn storerojn kaj la prilaborilo **efektivigas du aŭ tri** sinsekvajn legojn por havi la tutan instrukcion. La unuan okbiton, prenitan el la storo, oni lokas en la prilaborilan instrukciregistrumon kaj la postaj okbitoj eniras portempen storon; poste la prilaborilo daŭrigas la efektivigan fazon. Okaze de tia instrukckio oni parolas pri variablo-longo.

1.4. Adresregistrumo(j)

CPU povas utiligi registrumon aŭ registrumparon por enteni adreson de storero, kiun oni adresos. Se la adresregistrumo estas **programebla** (t.e., se ekzistas instrukcioj por ebligi al programisto modifi la enhavon de la registrumo), la programo povas konsistigi adreson en la adresregistrumon antaŭ ol efektivigi **storcelantan** instrukcion (instrukcio, kiu legas datumon el la storo, skribas datumon en la storon, aŭ operacias pri storita datumo).

1.5. Aritmetika/logika unuo (ALU)

Ĉiuj prilaboriloj enhavas artimetikan/logikan unuon, kiun oni ofte nomas nur ALU. ALU, kiel implicas ĝia nomo, estas parto de CPU kaj la ilo, kiu efektivigas aritmetikajn kaj logikajn operaciojn pri duumaj datumoj.

ALU devas enhavi **adiciilon**, kiu kapablas kombini la enhavojn de du registrumoj laŭ la logiko de la dubaza aritmetiko.

Tio permesas al la prilaborilo efektivigi aritmetikajn manipuladojn pri datumoj, kiujn ĝi prenas el la storo kaj el siaj aliaj enirejoj.

Utiligante nur la bazan adiciilon, sperta programisto kapablas skribi programojn, kiuj subtrahos, multiplikos kaj dividos, donante al la maŝino kompletajn aritmetikajn kapablojn. Tamen, praktike pli multaj ALUoj provizas aliajn enigitajn funkciojn, inklude peraparatan subtrahon, *Boole*-logikajn operaciojn kaj paŝigokapablojn.

ALU enhavas **flagbitojn**, kiuj spegulas iujn eventojn aŭ apartaĵojn, okazantajn en aritmetikaj kaj logikaj manipuladoj. Tial flagoj inkludas **kromon, nulon, signon** kaj **parecon.** Eblas programi saltojn, kiujn kondiĉos la stato de unu aŭ pluraj flagoj. Tiel ekzemple, la programo povas esti planita por salti al speciala rutina programo, se la krombito staras post adiciinstrukcio.

1.6. Regcirkvitaro

La regcirkvitaro estas la ĉefa funkciunuo en CPU. Uzante horloĝajn signalojn, la regvirkvitaro prizorgas la taŭgan sinsekvon de la eventoj, postulataj de ĉiu prilabora tasko. Leginte kaj malkodinte instrukcion, la regcirkvitaro eligas la adekvatajn signalojn (al unuoj de CPU aŭ internaj aŭ eksteraj por iniciati la taŭgan prilaboran agon. La regvirkvitaro ofte kapablas respondi al eksteraj signaloj, kiaj estas interrompo aŭ atendopeto. **Interrompopeto** estigas portempan ĉeson de la efektivigo de la ĉefa programo fare de la regcirkvitaro, salton al speciala rutina programo por servi la interrompitan aparaton, kaj poste reiron al la ĉefa programo. **Atendopeto** ofte devenas de stora aŭ eniga/eliga elemento, kiu operacias malpli rapide ol CPU. La regcirkvitaro atendigos CPUon ĝis kiam la storo aŭ la eniga/eliga pordo estos preta pri la datumo.

2. Komputoraj operacioj

Iuj operacioj estas bazaj por multaj komputoroj. Klara kompreno de tiuj bazaj operacioj estas necesa antaŭ ekzameno de la specifaj operacioj de aparta komputoro.

2.1. Tempado

La agadoj de komputoro estas ciklecaj. La prilaborilo legas instrukcion, efektivigas la postulatajn operaciojn, legas la postan instrukcion kaj tiel plu. Tiu orda sinsekvo de eventoj postulas precizan tempadon kaj CPU do bezonas libermovan oscilan horloĝon, kiu provizas referencon por ĉiuj prilaborilaj agoj. La duaĵo: lego kaj efektivigo de nur unu instrukcio estas nomata **instrukciciklo**. La parto de ciklo, asociita al klare difinita ago, estas nomata **stadio**. Kaj la intervalon inter pulsoj de tempada oscililo oni nomas **horloĝ-**

periodo. Ĝenerale unu aŭ pluraj horloĝperiodoj estas necesaj por la efektivigo de stadio kaj pluraj stadioj estas en unu ciklo.

2.2. Instrukcilego

La unua(j) stadio(j) de ĉiu instrukciciklo estas uzata(j) por legi la postan instrukcion. CPU eligas legsignalon kaj la enhavo de la instrukcimontrilo estas sendata al la storo, kiu respondas resendante la postan instrukcivorton. La unua okbito de la instrukcio eniras la instrukciregistrumon. Se la instrukcio konsistas el pli ol unu okbito, pliaj stadioj estas necesaj por havi ĉiujn okbitojn de la instrukcio. Kiam la tuta instrukcio estas en CPU, la instrukcimontrilon oni krementas (preparo por la posta instrukcilego) kaj oni malkodas la instrukcion. La operacio indikata de la instrukcio estas efektivigota dum la restaj stadioj de la instrukciciklo. La instrukcio povas postuli storlegon aŭ -skribon, enigon aŭ eligon kaj/aŭ internan CPU-operacion (ekz. interregistruma movo aŭ registrum-adicia operacio).

2.3 Storlego

Instrukcilego estas nur speciala storlega operacio, kiu movas la instrukcion en la CPUan instrukciregistrumon. La legita instrukcio tiam povas postuli datumlegon el la storo al CPU. CPU denove eligas legsignalon kaj sendas la ĝustan storadreson, la storo respondas, resendante la petitan vorton. La ricevita datumo eniras la akumulatoron aŭ unu el la aliaj ĝeneralcelaj registrumoj (ne la instrukciregistrumon).

2.4. Storskribo

La storskriba operacio similas la storlegan, escepte pri la direkto de la datumomovo. CPU eligas skribsignalon, sendas la ĝustan storadreson, kaj tiam sendas la datumvorton skribotan en la adresitan storeron.

2.5. Atendo (storsinĥronigo)

Kiel antaŭdirite, la agoj de la prilaborilo estas tempataj de mastra horloĝoscililo. La horloĝperiodo determinas la tempadon de ĉiuj prilaboraj agoj. Tamen la rapido de la prilaborila ciklo limiĝas pro la stora legotempo. Post

la sendo de la leg-adreso al la storo fare de la prilaborilo, ĝi devas atendi ĝis la respondo de la storo. Pli multaj storoj kapablas respondi multe pli rapide ol la laborciklo daŭras. Tamen kelkaj ne kapablas doni la adresitan okbiton antaŭ la minimuma tempo, indikata de la prilaborila horloĝo.

Prilaborilo do devus enhavi sinĥronigilon, kiu ebligus al la storo peti **atendostadion**. Kiam la storo ricevas validigitan leg- aŭ skribsignalon, ĝi metas petsignalon al la prilaborila enirejo PRETA, estigante provizoran paŭzon de CPU. Post la tempo necesa por respondi, la storo liberigas la prilaborilan enirejon PRETA kaj la instrukciciklo redaŭras.

2.6. Enigoj/Eligoj

Enigaj kaj eligaj operacioj similas al la storlegaj aŭ storskribaj operacioj, sed en tia okazo, periferia eniga/eliga aparato estas adresata anstataŭ storero. CPU eligas la taŭgan enigan aŭ eligan regsignalon, sendas la ĝustan aparatadreson kaj ricevas la eniran datumon, aŭ sendas la eliran.

Datumo povas eniri aŭ eliri laŭ paralela aŭ seria formo. Ĉiu datumo en karaktra komputoro estas prezentata laŭ duume kodita formo. Duuma datumvorto konsistas el bitaro, ĉiu bito estas aŭ unu aŭ nulo. **Paralela enigo/eligo** estas movo de ĉiuj vortbitoj samtempe, po unu bito laŭ unu drato. **Seria enigo/eligo** estas movo de unu bito post la alia laŭ nur unu drato. Kompreneble, la seria enigo/eligo estas multe malpli rapida, sed postulas ege malpli da materialo ol la paralela enigo/eligo.

2.7. Interrompoj

Interrompaj kapabloj ekzistas en multaj centraj prilaboriloj kiel rimedoj por plibonigi la prilaborilan efikon. Konsideru la okazon de komputoro, kiu prilaboras grandan kvanton da datumoj, kies partoj devos eliri al presilo. CPU kapablas eligi datuman okbiton en nur unu maŝinciklo, sed la presilo bezonas multajn maŝinciklojn por presi la karaktron, specifatan de la datuma okbito. CPU povus atendi, farante nenion, ĝis kiam la presilo estos preta akcepti la sekvantan datuman okbiton. Se en la komputoro estas disponebla la interrompa kapablo, CPU povas eligi okbiton kaj poste daŭrigi la datumprilaboron. Kiam la presilo estos preta akcepti la sekvantan okbiton, ĝi povus peti interrompon. Kiam CPU kvitancas la interrompon, ĝi haltigas la

efektivigon de la ĉefa programo kaj aŭtomate aktivigas la subrutinon, kiu elirigos la sekvantan okbiton. Post la okbita elirigo, CPU daŭrigas la efektivigon de la ĉefa programo. Notu, ke tio principe similas al subrutina voko, sed la salton oni iniciatas el ekstere, anstataŭ el la programo.

Ankaŭ pli kompleksaj interromp-strukturoj estas eblaj. En ili pluraj interrompaj aparatoj estas ligitaj al la sama prilaborilo, sed havas malsamajn prioritatajn nivelojn. Interrompebla prilaborado estas grava kapablo, kiu permesas maksimuman utilon de la aliaj prilaborilaj kapabloj por altgrada datumfluo en la sistemo.

2.8. Suspendo

Alia grava apartaĵo, kiu pligrandigas la datumfluon de prilaborilo, estas la **suspendo**. La suspendivo ebligas operaciojn laŭ rekta storatingo *(DMA)*.

En simplaj enigaj kaj eligaj operacioj, la prilaborilo mem regas la tutan datum-movon. La informo lokota en la storon estas movata de la eniga aparato al la prilaborilo kaj poste de la prilaborilo al la indikita storero. Same la informo, kiu iras de la storo al eliga aparato, iras tra la prilaborilo.

Tamen kelkaj periferiaj aparatoj kapablas movi informon al aŭ el la storo multe pli rapide, ol la prilaborilo mem kapablas fari. Se sufiĉe granda kvanto da datumoj devas esti movata al aŭ el tia aparato, tiam la datumfluo tra la sistemo pligrandiĝos, se la aparato mem povas efektivigi la movon. La prilaborilo devas provizore suspendi sian operaciadon dum tia movo por eviti konfliktojn, kiuj povus okazi, se la prilaborilo kaj la periferia aparato provus adresi la storon samtempe. Tial en kelkaj prilaboriloj estas disponebla la suspendivo.

3. Instrukciaro

Komputoro, eĉ se teĥnike altnivela, kapablas fari nur tion, kion oni ordonas al ĝi. Oni ordonas al komputoro per aro da koditaj instrukcioj nomata **programo**. La agokampo de la programisto estas nomata **programaro** (*En: software,* Fr: *logiciel*), opone al la **aparataro**, kiu konsistas el la reala komputora ekipaĵo. La komputora programaro konsistas el ĉiuj programoj skribitaj por tiu komputoro.

Kiam oni konceptas komputoron, la inĝenieroj kapabligas la centran prilaboran unuon (CPU) efektivigi apartan operaciaron. Oni planas CPUon tiel, ke specifa operacio efektiviĝas, kiam la CPUa reglogiko analizas apartan instrukcion. Konsekvence la operacio, kiun CPU povas efektivigi, difinas la komputoran **instrukciaron.**

Ĉiu komputora instrukcio ebligas al la programisto iniciati efektivigon de iu specifa operacio. Ĉiuj komputoroj enhavas iujn artimetikajn operaciojn en sia instrukciaro, ekz. instrukcion por adicii la enhavojn de du registrumoj. Ofte la instrukciaro inkludas la logikajn operaciojn (t.e. AŬi la enhavojn de du registrumoj) kaj registrum-operaciajn instrukciojn (t.e. krementi registrumon). Komputora instrukciaro enhavas ankaŭ instrukciojn, kiuj movas datumojn inter registrumoj, inter registrumo kaj storo, kaj inter registrumo kaj eniga/eliga aparato. Multaj instrukciaroj provizas ankaŭ **laŭkondiĉajn instrukciojn**. Laŭkondiĉa instrukcio specifas operacion efektivigotan nur, se iuj kondiĉoj estas plenumitaj; ekzemple, salto al aparta instrukcio, se la rezulto de la lasta operacio estas nulo. Laŭkondiĉaj instrukcioj provizas programon per decidiĝa kapablo.

La programisto, logike organizanta instrukcisekvencon laŭ kohera programo, kapablas ordoni al komputoro efektivigi tre specialan kaj utilan funkcion.

Tamen komputoro kapablas efektivigi nur programojn, kies instrukcioj estas laŭ duume kodita formo (t.e. vico de 1 kaj de 0), kiun oni nomas **maŝinkodo**. Ĉar estus treege malfacile programi en maŝinkodo, oni ellaboris programlingvojn. Ekzistas programoj, kiuj konvertas ilin en maŝinkodon, kiun kapablas interpreti la prilaborilo.

Asembla lingvo estas unu el la tipoj de programlingvoj. Unika asembla mnemoniko atribuiĝis al ĉiu el la komputaj instrukcioj. La programisto povas skribi programon (nomatan **fontprogramo**), utiligante mnemonikon kaj iujn perandojn; la fontprogramon oni poste konvertas en maŝininstrukciojn (nomatajn **celprogramo**). Ĉiun asemblan instrukcion oni konvertas al po unu maŝinkoda instrukcio (1 aŭ pluraj okbitoj) per asemblero. La asemblaj lingvoj kutime dependas de la maŝino (ili povas utili nur en unu tipo de komputoro).

La *8080*-a instrukciaro inkludas kvin malsamajn tipojn de instrukcioj:

— **Datum-mova grupo** — movas datumon inter registrumoj aŭ inter la storo kaj registrumo.

— **Aritmetika grupo** — adicias, subtrahas, krementas aŭ malkrementas datumon en registrumoj aŭ en la storo.

— **Logika grupo** — kaj, aŭ, eskluda aŭ, komparas, ringpaŝigas aŭ komplementas datumon en registrumoj aŭ en la storo.
— **Salta grupo** — laŭkondiĉaj kaj senkondiĉaj salt-instrukcioj, subrutinvokaj instrukcioj kaj reiraj instrukcioj.
— **Staka, eniga/eliga kaj maŝinrega grupo** — inkludas enigajn/eligajn instrukciojn, same kiel instrukciojn por utiligi la stakon kaj la internajn regflagojn.

3.1. Instrukciaj kaj datumaj formatoj

La storo de la *8080* organiziĝas laŭ okbitaj kvantoj nomataj okbitoj. Ĉiu okbito posedas unikan 16-bitan duuman adreson, kiu rilatas al ĝia laŭvica pozicio en la storo.

La *8080* kapablas rekte adresi ĝis 65 536 okbitojn de storo, kiu povas konsisti el nurlegebla storo (*Romo*) kaj el legebla modifebla storo (*RAM*).

En la *8080* oni storas datumon laŭ la formo de okbita duuma entjero.

Kiam registrumo aŭ datumvorto enhavas dubazan numeron, estas necese difini la ordon, laŭ kiu la bitoj de la numero skribiĝas. En *INTEL 8080* la biton 0 oni konsideras kiel la **malplej signifan biton** (MSB) kaj la biton 7 kiel la **plej signifan biton** (PSB).

La *8080*-aj program-instrukcioj povas longi unu, du aŭ tri okbitojn. Plurokbitaj instrukcioj devas esti storataj en sinsekvajn storerojn; la adreso de la unua okbito estas ĉiam uzata kiel la adreso de la instrukcio. La ĝusta instrukciformato dependas de la aparta farota operacio.

3.2. Adresaj manieroj

Ofte la datumo, koncernata de la instrukcio, estas storita en la storo. Kiam estas uzata plurokbita numera datumo, la datumo, same kiel la instrukcioj, estas storita en sinsekvaj storeroj, kun la malplej signifa okbito unue, sekvata de pli kaj pli signifaj okbitoj. La *8080* kapablas adresi datumon, storitan en la storo aŭ en registrumoj laŭ kvar malsamaj manieroj:
— **Rekta** — la okbitoj 2 kaj 3 de la instrukcio enhavas la ekzaktan storadreson de la datumo (la malplej signifaj bitoj de la adreso estas en la okbito 2, la plej signifaj bitoj estas en la okbito 3).
— **Registruma** — la instrukcio indikas la registrumon, kie troviĝas la datumo.

— **Traregistruma** — la instrukcio indikas registrum-paron, kiu enhavas la storadreson, kie troviĝas la datumo (la plej signifaj bitoj de la adreso estas en la unua registrumo de la paro, la malplej signifaj bitoj en la dua).
— **Tuja** — la instrukcio mem enhavas la datumon. Tio estas aŭ 8-bita kvanto aŭ 16-bita kvanto (la malpli signifa okbito estas la unua, la pli signifa okbito la dua).

La efektivigo de la instrukcioj progresas laŭ sinsekve plialtiĝantaj storeroj, escepte de la okazo, kiam la efektivigo estas direktata per interrompa aŭ salta instrukcio. Saltinstrukcio povas indiki la adreson de la tuj poste efektivigota instrukcio laŭ du manieroj:
— **Rekta** — la saltinstrukcio mem enhavas la adreson de la tuj poste efektivigota instrukcio. (Escepte de la instrukcio *RST*, la okbito 2 enhavas la malpli signifan adresparton kaj la okbito 3 la pli signifan adresparton).
— **Traregistruma** — la saltinstrukcio indikas registrumparon, kiu enhavas la adreson de la tuj poste efektivigota instrukcio. (La plej signifaj adresbitoj estas en la unua registrumo de la paro, la malplej signifaj bitoj en la dua).

La instrukcio *RST* estas speciala unuokbita vokinstrukcio (kutime uzata en interrompaj sekvencoj). *RST* inkludas 3-bitan kampon; la programregon ricevas tiu instrukcio, kies adreso valoras 8-oble la enhavon de la 3-bita kampo.

3.3. Kondiĉaj flagoj

En la *8080* estas kvin kondiĉaj flagoj, asociitaj al la efektiviĝo de instrukcio. Ili estas nulo, signo, pareco, kromo kaj mezkromo. Ĉiu estas reprezentata per aparta unubita registrumo en CPU. Flagon oni starigas, donante al la bito la valoron 1; flagon oni kuŝigas, donante al la bito la valoron 0.

Escepte de okazoj, kiam estas alie indikite, instrukcio tuŝas flagon jene:

Nulo: Se la rezulto de instrukcio valoras 0, ĝi starigas la flagon, alie ĝi kuŝigas ĝin.

Signo: Se la plej signifa bito de operacirezulto valoras 1, la flago staras, alie ĝi post la operacio kuŝas.

Pareco: Se la 2-modela sumo de la unu-valorantaj bitoj el la operacirezulto valoras 0 (t.e., se la rezulto estas para) la flago staras, alie ĝi kuŝas (t.e., se la rezulto estas nepara).

Kromo: Se la instrukcio estigas superfluon (de adicio) aŭ subfluon (de

subtraho aŭ komparo) ekster la plej signifa bito, la flago staras, alie ĝi kuŝas.

Mezkromo: Se la instrukcio kaŭzas superfluon ekster la bito 3 al la bito 4 en la rezulto, la mezkromo staras, alie ĝi kuŝas. Tiun flagon tuŝas adicioj en normala precizo, subtrahoj, krementoj, malkrementoj, komparoj kaj logikaj operacioj. Ĝi estas uzata precipe kun adicioj aŭ krementoj, kiuj antaŭas instrukcion *DAA* (dekbaza aligo de la akumulatoro).

3.4. Simboloj kaj mallongigoj

La sekvantaj simboloj kaj mallongigoj estas uzataj en la posta priskribo de la *8080*-instrukcioj.

Simboloj	**Signifoj**
akumulatoro	Registrumo A.
adr	16-bita adresa kvanto.
datumo	8-bita datuma kvanto.
d 16	16-bita datuma kvanto.
okbito 2	La dua okbito de la instrukcio (aŭ okb 2).
okbito 3	La tria okbito de la instrukcio (aŭ okb 3)
pordo	8-bita adreso de eniga/eliga aparato
r, rl, r2	Unu el lă registrumoj A,B,C,D,E,H,L.
CCC, FFF	La 3-bita nombro, indikanta unu el la registrumoj A,B,C,D,E,H,L (CCC = celo; FFF = fonto):

CCC aŭ FFF	**Registruma nomo**
111	A
000	B
001	C
010	D
011	E
100	H
101	L

rp — Unu el la registrumparoj:

B — reprezentas la paron B-C kun B kiel la pli signifa registrumo kaj C kiel la malpli signifa registrumo.

D — reprezentas la paron D-E kun D kiel la pli signifa registrumo kaj E kiel la malpli signifa registrumo.

H — reprezentas la paron H-L kun H kiel la pli signifa registrumo kaj L kiel la malpli signifa registrumo.

SP — reprezentas la 16-bitan stakmontran registrumon.

RP — La 2-bita nombro, indikanta unu el la registrumparoj B,D,H,SP:

RP	**Registrumparo**
00	B-C
01	D-E
10	H-L
11	SP

rs	La unua (pli signifa) registrumo el la indikita registrumparo.
rm	La dua (malpli signifa) registrumo el la indikita registrumparo.
PC	16-bita instrukcimontra registrumo (PCS kaj PCM estas uzataj por respektive indiki la pli signifan kaj la malpli signifan okbitojn).
SP	16-bita stakmontra registrumo (SPS kaj SPM estas uzataj por respektive indiki la pli signifan kaj la malpli signifan okbitojn).
ri	Bito "i" de la registrumo "r" (la bitoj numeratas de 7 ĝis 0 de maldekstre dekstren).
Z,S,P,C-Y,AC	La kondiĉaj flagoj: Z = nulo, S = signo, P = pareco, CY = kromo, AC = mezkromo.
( )	La enhavo de la storero aŭ registrumo meze de la krampoj.
←	"estas movata al" (dekstra parto maldekstren).
∧	Logika "KAJ"
⊻	Eskluda "AŬ"
∨	Inkluda "AŬ"
+	Adicio
×	Multipliko
↔	"estas interŝanĝata kun"
—	La unuuma komplemento [t.e. ($\bar{A}$)]
n	La restarta numero 0 ĝis 7
NNN.	La dubaza reprezento 000 ĝis 111 de la restarta numero 0 ĝis 7 respektive
M	La storo

4. Priskribo de la instrukcioj

4.1. Datum-mova grupo

Tiu instrukcigrupo movas datumon aŭ al aŭ el registrumo kaj storo. La kondiĉaj flagoj ne estas tuŝataj de ĉiuj instrukcioj de tiu grupo (tab. 1).

Tab. 1

Mnemoniko	Pervorta priskribo	Simbola priskribo	Kodo	Nombro de mašincikloj	Nombro de stadioj
MOV rl,r2	Movo el registrumo en registrumon.	(r)←(r2)	01CCCFFF	1	5
MOV M,r	Movo el registrumo en storeron (tra H-L)	((H) (L))←(r)	01110FFF	2	7
MOV r, M	Movo el storero (tra H-L) en registrumon.	(r)←((H) (L))	01CCC110	2	7
MVI r, datumo	Tuja movo de datumo en registrumon. La datumo estas en la instrukcio mem.	(r)←((H) (L))	00CCC110 datumo	2	7
LXI rp,d 16	Tuja movo de 16-bita datumo en registrumparon B-C aŭ SP. La malpli signifa datumparto estas en okbito 2.	(rs)←(okbito 3) (rm)←(okbito 2)	00RP0001 mpli s D pli s Dp	3	10
LDA adr	Rekta movo el storero en akumulatoron. La adreso de la storero estas en la instrukcio mem.	(A)←((okb 3) (okb 2))	00111010 mpli s A pli s Ap	4	13
STA adr	Rekta movo el akumulatoro en storeron. La adreso de la storero estas en la instrukcio mem.	((obk 3) (okb 2))←(A)	00110010 mpli pli s Ap	4	13
LHLD adr	Rekta movo (šarĝo) el storero en registrumparon H-L. La adreso de la storero estas en la instrukcio mem.	(L)←((okb 3) (okb2)) (H)←((okb 3) (okb 2) + 1))	00101010 mpli s A pli s Ap	5	16
SHLD adr	Rekta movo (storo) el registrumparo H-L en storeron. kie adreso estas en la instrukcio (malpli sign. parto de okbito 2)	((okb 3) (okb 2))←(L) ((okb 3) (okb 2) + 1)←H)	00100010 mpli s A pli s Ap	5	16
LDAX rp	Movo en akumulatoron el storero, kies adreso estas en registrumparo B-C aŭ D-E	(A)←((rp))	00RP1010	2	7
STAX rp	Movo el akumulatoro en la storeron. kies adreso estas en registrumparo B-C aŭ D-E	((rp))←(A)	00RP0010	2	7
XCHG	Interŝanĝo de la enhavoj de D-E kun K-L	(H)↔(D) (L)↔(E)	11101011	1	4

4.2. Aritmetika grupo

Tiu instrukcigrupo efektivigas aritmetikajn operaciojn per datumoj en registrumoj aŭ en la storo (tab. 2).

Escepte, kiam estas alie indikite, ĉiuj instrukcioj el tiu grupo tuŝas la nulan (Z), la signan (S), la parecan (P), la kromecan (CY) kaj la mezkromecan (AC) flagojn laŭ la kutimaj reguloj.

Ĉiuj subtrahaj operacioj efektivigas laŭ aritmetiko je duuma komplemento, starigas la flagon kromo (CY) por indiki subnuliĝon kaj kuŝigas ĝin por indiki nenian subnuliĝon.

Tab. 2

Mnemoniko	Pervorta priskribo	Simbola priskribo	Kodo	Nombro de mašincikloj	Nombro de stadioj
ADD r	Adicio de registrumo en akumulatoron.	(A)←(A)+(r)	10000FFF	1	4
ADD M	Adicio de storero, kies adreso estas en H-L, en akumulatoron.	(A)←(A)+((H) (L))	10000110	2	7
ADI	Tuja adicio de datumo en akumulatoron. La datumo estas en la instrukcio mem.	(A)←(A)+(okb 2)	11000110 datumo	2	7
ADC r	Adicio de registrumo plus kromflago en A	(A)←(A)+(r)+(CY)	10001FFF	1	4
ADC M	Adicio de storero, kies adreso estas en H-L, plus kromflago (CY) en akumulatoron	(A)←(A)+((H) (L))+(CY)	10001110	27	
ACI datumo	Tuja adicio de datumo plus kromflago plus akumulatoro en akumulatoron.	(A)←(A)+(okb 2)+(CY)	11001110 datumo	2	7
SUB r	Subtraho de registrumo el akumulatoro.	(A)←(A)-(r)	10010FFF	1	4
SUB M	Subtraho de storero, kies adreso estas en H-L, el akumulatoro.	(A)←(A)-((H) (L))	10010110	2	7
SUI datumo	Tuja subtraho de datumo el akumulatoro. La datumo estas en la instrukcio mem.	(A)←(A)-(okbito 2)	11010110 datumo	2	7
SBB r	Subtraho de registrumo kaj de kromflago (CY) el akumulatoro.	(A)←(A)-(r)-(CY)	10011FFF	1	4
SBB M	Subtraho de storero, kies adreso estas en H-L, kaj de kromflago el akumulatoro.	(A)←(A)-((H) (L))-(CY)	10011110	2	7
SBI datumo	Tuja subtraho de datumo kaj de kromflago (CY) el akumulatoro.	(A)←(A)-(okb 2)-(CY)	11011110 datumo	2	7

Daŭrigo de Tab. 2

INR r	Inkremento je unu de la enhavo de la registrumo r. Nur (CY) ne tuŝata.	(r)←(r)+1	00CCC100	1	5
INR M	Inkremento je unu de la enhavo de la stokero, kies adreso estas en H-L.	((h) (L))←((H) (L))+1	00110100	3	10
DCR r	Dekremento je unu de enhavo de la registrumo r. Nur (CY) ne tuŝata.	(r)←(r)-1	00CCC101	1	5
DCR M	Dekremento je unu de la enhavo de la stokero, kies adreso estas en H-L.	((h) (L)). ←((H) (l))-1	00110101	3	10
INX rp	Inkremento je unu de la enhavo de la registrumparo rp. Neniu flago tuŝata.	(rs) (rm)←(rs) (rm)+1	00RP0011	1	5
DCX rp	Dekremento je unu de la enhavo de la registrumparo rp. Neniu flago tuŝata.	(rs) (rm)←(rs) (rm)-1	00RP1011	1	5
DAD rp	Adicio de la enhavo de la registrumparo rp (B-C, D-E, H-L aŭ SP) al la enhavo de registrumparo H-L. Nur la kromflago (CY) estas tuŝata.	(H l)←(h) (L)+(rs) (rm)	00RP1001	3	10
DAA	Dekbaza alformado de la akumulatoro. La 8-bita nombro en la akumulatoro estas modifata por formi du 4-bitajn BCDajn ciferojn laŭ la sekvanta procedo: 1. Se la valoro de la malplej signifaj 4 bitoj de la akumulatoro superas 9, aŭ se la mezkroma flago (AC) staras, oni adicias 6 al la akumulatoro. 2. Se la valoro de la plej signifaj 4 bitoj de la akumulatoro nun superas 9, aŭ se la kromflago (CY) staras, oni adicias 6 al plej signifaj 4 bitoj de la akumulatoro. Notu: ĉiuj flagoj estas tuŝataj.	00100111	1	4	

4.3. Logika grupo

Tiu instrukcigrupo efektivigas logikajn (boolajn, de la nomo *Boole*, operaciojn kun datumoj en registrumoj aŭ en la storo kaj kun kondiĉaj flagoj (tab. 3).

Escepte de okazoj alie indikitaj, ĉiuj instrukcioj en tiu grupo tuŝas la flagojn Z (nulo), P (pareco), CY (kromo), AC (mezkromo), S (signo) laŭ la kutimaj reguloj.

Tab. 3

Mnemoniko	Pervorta priskribo	Simbola priskribo	Kodo	Nombro de mašincikloj	Nombro de stadioj
ANA r	Logika KAJo de la registrumo r al la akumulatoro. Ĝi kušigas la kromflagon (CY).	(A)←(A)∧(r)	10100FFF	1	4
ANA M	Logika KAJo de la stokero, kies adreso estas en H-L, al la akumulatoro. Ĝi kušigas la kromflagon (CY).	(A)←(A)∧((H) (L))	10100110	2	7
ANI datumo	Tuja logika KAJo de la datumo al la akumulatoro. Ĝi kušigas la flagojn kromo (CY) kaj mezkromo (AC).	(A)←(A)∧(0kbito 2)	11100110 datumo	2	7
XRA r	Logika eskluda AŬo de la registrumo R al la akumulatoro. Ĝi kušigas la flagojn kromo (CY) kaj mezkromo (AC).	(A)←(A)∨) (r)	10101FFF	1	4
XRA M	Logika eskluda AŬo de la stokero, kies adreso estas en H-L, al la akumulatoro. Ĝi kušigas la flagojn kromo (CY), mezkromo (AC).	(A)←(A)∨((H) (L))	10101110	2	7
XRI datumo	Tuja logika eskluda AŬo de la datumo al la akumulatoro. Ĝi kušigas la flagojn kromo (CY) kaj mezkromo (AC).	(A)←(A)∨(okbito 2)	11101110 datumo	2	7
ORA r	Logika inkluda AŬo de la registrumo r al la akumulatoro. Ĝi kušigas la flagojn kromo (CY) kaj mezkromo (AC).	(A)←(A)∨(r)	10110FFF	1	4
ORA M	Logika inkluda AŬo de la stokero, kies adreso estas en H-L, al la akumulatoro. Ĝi kušigas la flagojn (CY) kaj (AC).	(A)←∨((H) (L))	10110110	2	7
ORI datumo	Tuja logika inkluda AŬo de la datumo al la akumulatoro. Ĝi kušigas la flagojn kromo (CY) kaj mezkromo (AC).	(A)←∨(0kbito 2)	11110110 datumo	2	7
CMP r	Komparo de registrumo al la akumulatoro. Tiu instrukcio ne modifas la registrumojn. (Z) staras, se (A)=(r); (CY) staras, se (A)<(r).	(A)−(r)	10111FFF	1	4
CMP M	Komparo de stokero, kies adreso estas en H-L, al la akumulatoro. Flagoj kiel ĉi-supre.	(A)−((H) (L))	10111110	2	7
CPI datumo	Tuja komparo de datumo al la akumulatoro. La datumo estas en la instrukcio mem.	(A)−(okbito 2)	11111110 datumo	2	7
RLC	Maldekstra ringpašigo. Nur la kromflago (CY) estas tuŝata.	$(A_{i+1})\leftarrow(A_i)$ $(A_0)\leftarrow(A_7)$, (CY)←(A_7)	00000111	1	4
RRC	Dekstra ringpašigo. Nur la kromflago (CY) estas tuŝata.	$(A_i)\leftarrow(A_{i+1})$ $(A_7)\leftarrow(A_0)$, (CY)←(A_0)	00001111	1	4

Daŭrigo de Tab. 3

RAL	Maldekstra ringpaŝigo tra la kromflago (CY) Nur la kromflago (CY) estas tuŝata. La Malgrava bito kaj la flago (CY) post la paŝigo valoras la enhavon de la grava bito antaŭ la paŝigo.	$(A_{n+1}) \leftarrow (A_n)$ $(CY) \leftarrow (A_7)$, $(A_0) \leftarrow (CY)$	00010111	1	4
RAR	Dekstra ringpaŝigo tra la kromflago (CY) Nur la kromflago (CY) estas tuŝata	$(A_n) \leftarrow (A_{n+1})$ $(CY) \leftarrow (A_0)$, $(A_7) \leftarrow (CY)$	00011111	1	4
CMA	Komplementado de la akumulatoro. Bitoj 0 tarigas 1 kaj inverse. La flagoj ne estas tuŝataj.	$(A) \leftarrow (\overline{A})$	00101111	1	4
CMC	Komplementado de la kromflago (CY). Nur la kromflago (CY) estas tuŝata.	$(CY) \leftarrow (\overline{CY})$	00111111	1	4
STC	Starigo de la kromflago (CY). Nur la kromflago (CY) estas tuŝata.	$(CY) \leftarrow 1$	00110111	1	4

4.4. Salta grupo

Tiu instrukcigrupo interrompas la normalan sinsekvan efektivigon de la programo (tab. 4).

Neniu instrukcio de tiu grupo modifas la kondiĉajn flagojn.

La salt-instrukcioj estas aŭ senkondiĉaj aŭ laŭkondiĉaj. La senkondiĉaj saltoj efektivigas nur la indikitan operacion pri registrumo PC (la instrukcimontrilo). La laŭkondiĉaj saltoj analizas la staton de unu el la kvar prilaborilaj flagoj por decidi, ĉu ĝi devas efektivigi la indikitan salton. La kondiĉoj, kiujn oni povas indiki, estas sekvantaj:

Kondiĉo	KKK		
NZ — nenula (Z = 0)	000	PE — para (P = 1)	101
NC — senkroma (CY = 0)	010	P — pozitiva aŭ nula (S = 0)	110
C — kunkroma (CY = 1)	011	M — negativa (S = 1)	111
PO — nepara (P = 0)	100		

4.5. Staka, eniga/eliga kaj maŝinrega grupo

Tiu instrukcigrupo efektivigas enigojn aŭ eligojn, utiligas la stakon kaj tuŝas la internajn kondiĉajn flagojn (nomatajn ankaŭ: regflagoj) (tab. 5).

Escepte de okazoj, kiam estas alie indikite, neniu instrukcio de tiu grupo tuŝas la kondiĉajn flagojn.

Tab. 4

Mnemoniko	Pervorta priskribo	Simbola priskribo	Kodo	Nombro de maŝincikloj	Nombro de stadioj
JMP adr	Senkondiĉa salto al la indikita adreso en la instrukcio mem: la malpli signifa adresparto estas en la okbito 2.	(PC) ← (Okb 3) (okb 2)	11000011 mpli s A pli s Ap	3	10
JC adr	Salto, se kromflago valoras 1((CY) = 1), al la adreso indikita en la instrukcio; t.e. salto, se okazis super- aŭ subfluon.	se (CY) = 1 do (PC) ← (Okb 3) (Okb 2)	11011010 mpli s A pli s Ap	3	10
JNC adr	Salto, se kromflago valoras 0((CY) = 0), al la adreso indikita en la instrukcio; t.e. salto, se ne okazis superfluo.	se (CY) = 0 do (Pc) ← (Okb 3) (Okb 2)	11010010 mpli s A pli s Ap	3	10
SZ adr	Salto, se nulflago valoras 1(Z = 1), al la adreso indikita en la instrukcio; t.e salto, se la rezulto estas nulo.	se (Z) = 1 do (PC) ← (Okb 3) (Okb 2)	11001010 mpli s A pli s Ap	3	10
JNZ adr	Salto, se nulflago valoras 0 (Z = 0), al la adreso indikita en la instrukcio; t.e. salto, se la rezulto ne estas nulo.	se (Z) = 0 do (PC) ← (Okb 3) (Okb 2)	11000010 mpli s A pli s Ap	3	10
JP adr	Salto, se rezulto estas pozitiva aŭ nula (S = 0), al la adreso indikita en la instrukcio.	se (S = 0) do (PC) ← (Okb 3) (Okb 2)	11110010 mpli s A pli s Ap	3	10
JM adr	Salto, se rezulto estas nepara (P = 0), al la adreso indikita en la instrukcio	se (S = 1) do (PC) ← (Okb 3) (Okb 2)	11111010 mpli s A pli s Ap	3	10
JPE	Salto, se rezulto estas neparaca (P = 0), al la adreso indikita en la instrukcio	se (P = 0) do (PC) ← (Okb 3) (Okb 2)	11101010 mpli s A pli s Ap	3	10
JPO adr	Salto, se rezulto estas para (P = 1), al la adreso indikita en la instrukcio	se (P = 1) do (PC) ← (Okb 3) (Okb 2)	11100010 mpli s A pli s Ap	3	10
CALL adr	Senkondiĉa voko al la adreso indikita en la instrukcio, pro savo de la enhavo de instrukcimontrilo (SP) en la stakon, antaŭ la salto, por la reveno.	((SP)-1) ← (PCS) ((SP)-2) ← (PCM) (SP) ← (SP)-2 (PC) ← (Okb 3) (Okb 2)	11001101 mpli s A pli s Ap	5	17
CC adr	Voko, se la kromflago valoras 1 (CY = 1); t.e. voko, se okazis dum la lasta operacio superfluo aŭ subfluo.	se (CY = 1) do same kiel CALL	11011100 mpli s A pli s Ap	3 5	11 17
CNC adr	Voko, se la kromflago kuŝas (CY = 0); t.e. voko, se ne okazis dum la lasta operacio superfluo aŭ subfluo.	se (CY = 0) do same kiel CALL	11010100 mpli s A pli s Ap	3 5	11 17

Daŭrigo de Tab. 4

CZ adr	Voko, se nulflago staras (Z = 1); t.e. voko, se la rezulto estas nulo	se (Z = 1) do same kiel CALL	11001100 mpli s A pli s Ap	3 5	11 17
CNZ adr	Voko, se la nulflago kušas (Z = 0); t.e. voko, se la rezulto ne estas nulo	se (Z = 0) do same kiel CALL	11000100 mpli s A pli s Ap	3 5	11 17
CP adr	Voko, se rezulto estas pozitiva aŭ nula (S = 0), al la adreso indikita en la instrukcio.	se (S = 0) do same kiel CALL	11110100 mpli s A pli s A	3 5	11 17
CM adr	Voko, se rezulto estas negativa (S = 1), al la adreso indikita en la instrukcio	se (S = 1) do same kiel CALL	11111100 mpli s A pli s Ap	3 5	11 17
CPE adr	Voko, se rezulto estas para (P = 0), al la adreso indikita en la instrukcio.	se (P = 0) do same kiel CALL	11101100 mpli s A pli s A	3 5	11 17
CPO adr	Voko, se rezulto estas para (P = 1), al la adreso indikita en la instrukcio	se (P = 1) do same kiel CALL	11100100 mpli s A pli s Ap	3 5	11 17
RET	Senkondiča reiro. La adreso de la posta instrukcio estas en la stako.	(PCM7 ← ((SP)) (PCS) ← ((SP) + 1) (SP) ← (SP) + 2	11001001	3	10
RC	Reiro, se la kromflago staras (CY = 1)	se (CY = 1) do same kiel RET	11011000	1 3	5 11
RNC	Reiro, se la kromflago kušas (CY = 0)	se (CY = 0) do same kiel RET	11010000	1 3	5 11
RZ	Reiro, se la nulflago staras (Z = 1); t.e. se la rezulto estas nulo	se (Z = 1) do same kiel RET	11001000	1 3	5 11
RNZ	Reiro, se la nulflago kušas (Z = 0); t.e. reiro se la rezulto ne estas nulo.	se (Z = 0) do same kiel RET	11000000	1 3	5 11
RP	Reiro, se rezulto estas pozitiva aŭ nula (S = 0), al la adreso savita en la stakon	se (S = 0) do same kiel RET	11110000	1 3	5 11
RM	Reiro, se rezulto estas negativa (S = 1), al la adreso savita en la stakon	se (S = 1) do same kiel RET	11111000	1 3	5 11
RPE	Reiro, se rezulto estas nepara (P = 0), al la adreso savita en la stakon	se (P = 0) do same kiel RET	11101000	1 3	5 11
RPO	Reiro, se rezulto estas para (P = 1), al la adreso savita en la stakon.	se (P = 1) do same kiel RET	11100000	1 3	5 11
RST n	Restarto je adreso 8xn (n = 0 ĝis 7). Ĝi funkcias kiel CALL 8xn. Enhavo de la instrukcimontrilo post la instrukcio: 0000 0000 00NN N000.	((SP)-1) ← (PCS) ((SP)-2) ← (PCM) (SP) ← (SP)-2 (PC) ← 8 x (NNN)	11NNN111	3	11
PCHL	Movo de la registrumparo H-L en la registrumon PC (instrukcimontrilo).	(PCS) ← (H) (PCM) ← (L)	11101001	1	5

Tab. 5

Mnemoniko	Pervorta priskribo	Simbola priskribo	Kodo	Nombro de mašinciklo	Nombro de stadioj
PUSH rp	Puŝo de registrumparo en la stakon. *Noto*: La registrumparo rp = SP ne devas esti indikata.	(SP)-1 ← (rs) ((SP)-2) ← (rm) (SP) ← (SP)-2	11RP0101	3	11
PUSH PSW	Puŝo de la akumulatoro kaj de la kondičaj flagoj en la stakon.	((SP)-1) ← (A) ((SP)-2)$_0$ ← (CY) ((SP)-2)$_1$ ← 1 ((SP)-2)$_2$ ← (P) ((SP)-2)$_3$ ← (0) ((SP)-2)$_4$ ← (AC) ((SP)-2)$_5$ ← 0 ((SP)-2)$_6$ ← (Z) ((SP)-2)$_7$ ← (S) (SP) ← (SP)-2	11110101	3	11
	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 S Z 0 AC 0 P 1 CY : Flagvorto Signoflago en la plej signifa bito (S). Nulflago (Z) en la bito D_6. Mezkromflago (AC) en la bito D_4. Parecflago (P) en la bito D_2. Kromflago (CY) en la bito D_0.				
POP rp	Tiro el la stako en registrumparon rp. *Noto*: la registrumparo rp = SP ne devas esti indikata.	(rm) ← ((SP)) (rs) ← ((SP) + 1) (SP) ← (SP) + 2	11RP0001	3	10
POP PSW	Tiro el la stako en la akumulatoron kaj en la kondičajn flagojn. Ĉiuj flagoj, kompreneble, estas tušataj de tiu instrukcio.	(CY) ← ((SP))$_0$, (P) ← ((SP))$_2$ (AC) ← ((SP))$_4$, (Z) ← ((SP))$_6$ (S) ← ((SP))$_7$, (A) ← ((SP) + 1) (SP) ← (SP) + 2	11110001	3	10
XTHL	Interŝanĝo inter la staksupro kaj la registrumparo H-L.	(H) ← ((SP)) (H) ↔ ((SP) + 1)	11100011	5	18
SPHL	Movo de registrumparo H-L en la registrumon SP (stakmontrilo).	(SP) ← (H) (L)	11111001	1	5
OUT pordo	Eligo de datumo el la akumulatoro tra pordo indikita en la instrukcio	(datumo) ← (A)	11010011 pordo	3	10
EI	Ebligo de interrompoj. Tuj post la efektiviĝo de tiu instrukcio, la interrompoj plu ne estos akceptataj ĝis la efektiviĝo de instrukcio, kiu tuj sekvos la instrukcion EI.		11110011	1	4
HLT	Ĝi haltigas la prilaborilon je la adreso de tiu instrukcio. La registrumoj kaj la kondičaj flagoj restas netušitaj.		01110110	1	7
NOP	Nenioperacio. Tiu instrukcio nur progresigas la instrukcimontrilon (SP). Ĝi ne tušas la registrumojn nek la kondičajn flagojn.		00000000	1	4

Tab. 6

MOVO

40 MOV B,B	68 MOV L,B
41 MOV B,C	69 MOV L,C
42 MOV B,D	6A MOV L,D
43 MOV B,E	6B MOV L,E
44 MOV B,H	6C MOV L,H
45 MOV B,L	6D MOV L,L
46 MOV B,M	6E MOV L,M
47 MOV B,A	6F MOV L,A
48 MOV C,B	70 MOV M,B
49 MOV C,C	71 MOV M,C
4A MOV C,D	72 MOV M,D
4B MOV C,E	73 MOV M,E
4C MOV C,H	74 MOV M,H
4D MOV C,L	75 MOV M,L
4E MOV C,M	—
4F MOV C,A	77 MOV A,A
50 MOV D,B	78 MOV A,B
51 MOV D,C	79 MOV A,C
52 MOV D,D	7A MOV A,D
53 MOV D,E	7B MOV A,E
54 MOV D,H	7C MOV A,H
55 MOV D,L	7D MOV A,L
56 MOV D,M	7E MOV A,M
57 MOV D,A	7F MOV A,A
58 MOV E,B	
59 MOV E,C	
5A MOV E,D	
5B MOV E,E	
5C MOV E,H	
5D MOV E,L	
5E MOV E,M	
5F MOV E,A	
60 MOV H,B	
61 MOV H,C	
62 MOV H,D	
63 MOV H,E	
64 MOV H,H	
65 MOV H,L	
66 MOV H,M	
67 MOV H,A	

STAKAJ OPERACIOJ

C5 PUSH B,B
D5 PUSH B,D
E5 PUSH B,H
F5 PUSH B,PSW

C1 POP B,B
D1 POP B,F
E1 POP B,H
F1 POP B,PSW ●

E3 XTHL
F9 SPHL

KUN LA AKUMULATORO ●

80 ADD B,B	A8 XRA B,B
81 ADD B,C	A9 XRA B,C
82 ADD B,D	AA XRA B,D
83 ADD B,E	AB XRA B,E
84 ADD B,H	AC XRA B,H
85 ADD B,L	AD XRA B,L
86 ADD B,M	AE XRA B,M
87 ADD B,A	AF XRA B,A
88 ADD B,B	B0 ORA B,B
89 ADD B,C	B1 ORA B,C
8A ADD B,D	B2 ORA B,D
8B ADD B,E	B3 ORA B,E
8C ADD B,H	B4 ORA B,H
8D ADD B,L	B5 ORA B,L
8E ADD B,M	B6 ORA B,M
8F ADD B,S	B7 ORA B,A
90 SUB B,B	B8 CMP B,B
91 SUB B,C	B9 CMP B,C
92 SUB B,D	BA CMP B,D
93 SUB B,E	BB CMP B,E
94 SUB B,H	BC CMP B,H
95 SUB B,L	BD CMP B,L
96 SUB B,M	BE CMP B,M
97 SUB B,S	BF CMP B,A
98 SBB B,B	
99 SBB B,C	
9A SBB B,D	
9B SBB B,E	
9C SBB B,H	
9D SBB B,L	
9E SBB B,M	
9F SBB B,A	
A0 ANA B,B	
A1 ANA B,C	
A2 ANA B,D	
A3 ANA B,E	
A4 ANA B,H	
A5 ANA B,L	
A6 ANA B,M	
A7 ANA B,A	

RINGPAŜIGO ▲

07 RLC
0F RRC
17 RAL
1F RAR

SPECIALAJ

EB XCHG
27 DAA ●
2F CMA
37 STC ▲
3F CMC ▲

SALTO

C3 JMP
C2 JNZ
CA JZ
D2 JC
DA JC
E2 JPE
EA JPE
F2 JP
FA JM
E9 PCHL

Adr

TUJA ŜARGO

01 LXI B,
11 LXI D,
21 LXI H,
31 LXI SP,

Adr

DUOBLA ADICIO ▲

09 DAD B
19 DAD D
29 DAD H
39 DAD SP

ŜARGO STORO

0A LDAX B
1A LDAX D
2A LHLD Adr
3A LDA Adr

02 STAX B
12 STAX D
22 SHLD Adr
32 STA Adr

VOKO

CD CALL
C4 CNZ
CC CZ
D4 CNC
DC CC
E4 CPO
EC CPE
F4 CP
FC CM

Adr

REIRO

C9 RET
C0 RNZ
C8 RZ
C8 RZ
D0 RNC
D8 RC
E0 RPO
F0 RP
F8 RM

RESTARTO

C7 RST 0
CF RST 1
D7 RST 2
DF RST 3
E7 RST 4
EF RST 5
F7 RST 6
FF RST 7

TUJA MOVO

06 MVI B,
0E MVI C,
16 MVI D,
1E MVI E,
26 MVI H,
2E MVI L,
36 MVI M,
3E MVI A,

D8

TUJA AL AKUMULATORO ●

C6 ADI
CE ACI
D6 SUI
DE SBI
E6 ANI
EE XRI
F6 ORI
FE CPI

D8

REGO

00 NOP
76 HLT
F3 DI
FB EI

ENIGO/ELIGO

D3 OUT
DB IN

D8

INKREMENTO ▽

04 INR B
0C INR C
14 INR D
1C INR E
24 INR H
2C INR L
34 INR M
3C INR A

03 INX B
13 INX D
23 INX H
33 INX SP

DEKREMENTO ▽

05 DCR B
0D0 DCR C
15 DCR D
1D DCR E
25 DCR H
2D DCR L
35 DCR M
3D DCR A

0B DCX B
1B DCX D
2B DCX H
3B DCX SP

PSEŬDA INSTRUKCIO

ORG Adr
END
EQU D16
SET D16
DS D16
DB D8, [D8]*
DW D16 [D16]*

IF D16
ENDIF

MACRO | |
ENDM

KONSTANT-DIFINOJ

0BDH, 1AH } 16 uma
105D, 105 } 10 uma
720, 720 } 8 uma
11011B, 00110B } 1 uma
'TEST', 'A', 'B' } ASCII

OPERATOROJ

( )
x, /, MOD, SHL, SHR
+, -
NOT
AND
OR, XOR

IMPLICAJ ASIGNOJ

A	SET 7
B	SET 0
C	SET 1
D	SET 2
E	SET 3
H	SET 4
L	SET 5
M	SET 6
SP	SET 6
SPW	SET 6

FLAGOKBITA STAKFORMATO

7	6	5	4	3	2	1	0
S	Z	0	AC	0	P	1	C

● = ĉiuj flagoj (CY,Z,S,P,AC) tuŝataj
▲ = nur la kromflago (CY) tuŝata
▽ = ĉiuj flagoj, escepte de la kromflago (CY), tuŝataj; (esceptoj: INX, DCX tuŝas neniun flagon) *ASCII* = Usona Norma Kodo por Inform-Interŝanĝo.

Daŭrigo de Tab. 6

00 NOP	28	50 MOV D,B	78 MOV A,B	A0 ANA B	C8 RZ	F0 RP
01 LXI B,D16	29 DAD H	51 MOV D,C	79 MOV A,C	A1 ANA C	C9 RET Adr	F1 POP PSW
02 STAX B	2A LHLD Adr	52 MOV M,D	7A MOV A,D	A2 ANA D	CA JZ Adr	F2 JP Adr
03 INX B	2B DCX H	53 MOV D,E	7B MOV A,E	A3 ANA E	CB	F3 DI
04 INR B	2C INR L	54 MOV D,H	7C MOV A,H	A4 ANA H	CC CZ Adr	F4 CP Adr
05 DCR B	2D DCR L	55 MOV D,L	7D MOV A,L	A5 ANA L	CD CALL Adr	F5 PUSH PSW
06 MVI B,D8	2E MVI L,D8	56 MOV D,M	7E MOV A,M	A6 ANA M	CE ACI D8	F6 ORI D8
07 RLC	2F CMA	57 MOV D,A	7F MOV A,A	A7 ANA A	CF RST 1	F7 RST 6
08	30	58 MOV E,B	80 ADD D	A8 XRA B	D0 RNC	F8 RM
09 DAD B	31 LXI SP,D16	59 MOV E,C	81 ADD C	A9 XRA C	D1 POP D	F9 SPHL
0A LDAX B	32 STA Adr	5A MOV E,D	82 ADD D	AA ZRA D	D2 JNC Adr	FA JM Adr
0B DCX B	33 INX SP	5B MOV E,E	83 ADD E	AB ZRA E	D3 OUT D8	FB EI
0C INR C	34 INR M	5C MOV E,H	84 ADD H	AC XRA H	D4 CNC Adr	FC CM Adr
0D DCR C	35 DCR M	5D MOV E,L	85 ADD L	AD XRA L	D5 PUSCH D	FD
0E MVI C,D8	36 MVI M,D8	5E MOV E,M	86 ADD M	AE XRA M	D6 SUI D8	FE CPI D8
0F RRC	37 STC	EF MOV E,A	87 ADD A	AF XRA A	D7 RST 2	FF RST 7
10	38	60 MOV H,B	88 ADC B	B0 ORA B	D8 RC	
11 LXI D,D16	39 DAD SP	61 MOV H,C	89 ADC C	B1 ORA C	D9	
12 STAX D	3A LDA Adr	62 MOV H,D	8A ADC D	B2 ORA D	DA JC Adr	
13 INX D	3B DCX SP	63 MOV H,F	8B ADC E	B3 ORA E	DB IN D8	
14 INR D	3C INR A	64 MOV H,H	8C ADC H	B4 ORA H	DC CC Adr	
15 DCR D	3D DCR A	65 MOV H,L	8D ADC L	B5 ORA L	DD	
16 MVI D,D8	3E MVI A,D8	66 MOV H,M	8E ADC M	B6 ORA M	DE SBI D8	
17 RAL	3F CMC	67 MOV H,A	8F ADC A	B7 ORA A	DF RST 3	
18	40 MOV B,B	68 MOV L,B	90 SUB B	B8 CMP B	E0 RPO	
19 DAD D	41 MOV B,C	69 MOV L,C	91 SUB C	B9 CMP C	E1 POP H	
1A LDAZ D	42 MOV B,D	6A MOV L,D	92 SUB D	BA CMP D	E2 JPO Adr	
1B DCX D	43 MOV B,E	6B MOV L,E	93 SUB E	BB CMP E	E3 XTHL	
1C INR E	44 MOV B,H	6C MOV L,H	94 SUB H	BC CMP H	E4 CPO Adr	
1D DCR E	45 MOV B,L	6D MOV L,L	95 SUB L	BD CMP L	E5 PUSH H	
1E MVI E, D8	46 MOV B,A	6E MOV L,M	96 SUB M	BE CMP M	E6 ANI D8	
1F RAR	47 MOV B,A	6F MOV L,A	97 SUB A	BF CMP A	E7 RST 4	
20	48 MOV C,B	70 MOV M,B	98 SBB B	C0 RNZ	E8 RPE	
21 LXI H,D16	49 MOV C,C	71 MOV M,C	99 SBB C	C1 POP B	E9 PCHL	
22 SHLD	4A MOV C,D	72 MOV M,D	9A SBB D	C2 JNZ Adr	EA JPE Adr	
23 INX H	4B MOV C,E	73 MOV M,E	9B SBB E	C3 JMP Adr	EB XCHG	
24 INR H	4C MOV C,H	74 MOV M,H	9C SBB H	C4 CNZ Adr	EC CPE Adr	
25 DCR H	4D MOV C,L	75 MOV M,L	9D SBB L	C5 PUSH B	ED	
26 MVI H,D8	4E MOV C,M	76 HLT	9E SBB M	C6 ADI D8	EE XRI D8	
27 DAA	4F MOV C,A	77 MOV M,A	9F SBB A	C7 RST 0	EF RST 5	

D8 = konstanto aŭ logika/aritmetika esprimo, kiu kalkuliĝas en 8-bitan datum-kvanton.
Adr = 16-bita adreso
D16 = konstanto aŭ logika/aritmetika esprimo, kiu kalkuliĝas en 16-bitan datumkvanton
(D8)* aŭ (D16)* = listo, eble vaka, kun tiaj esprimoj

Tab. 7. Teletajpila kodo *ASCII* (Usona Norma Kodo por Inform-Interŝanĝo).

16-uma kodo	Mnemoniko aŭ karakteo	Bendtruoj 8	7	6	5	4	3	2	1	Teletajpilaj klavoj	Rimarkigoj
00	NUL	□	□	□	□	□	□	□	□	CTRL SHIFT	Plenigo
81	SOH	■	□	□	□	□	□	□	■	CTRL A	Kapumkomenco
82	STX	■	□	□	□	□	□	■	□	CTRL B	Tekstofino
03	ETX	□	□	□	□	□	□	■	■	CTRL C	Tramsmisifino
84	EOT	■	□	□	□	□	■	□	□	CTRL D	Demando
05	ENQ	□	□	□	□	□	■	□	■	CTRL E	Pozitiva kvitanco
06	ACK	□	□	□	□	□	■	■	□	CTRL F	Sonoro
87	BEL	■	□	□	□	□	■	■	■	CTRL G	
88	BS	■	□	□	□	■	■	□	□	CTRL H	Retroa spaceto
09	HT	□	□	□	□	■	□	□	■	CTRL I	Horizontala tabelado
0A	LF	□	□	□	□	■	□	■	□	Line feed	Linisalto
8B	VT	■	□	□	□	■	□	■	■	CTRL K	Vertikala tabelado
0C	FF	□	□	□	□	■	■	□	□	CTRL L	Formula prezento aŭ paĝsalto
8D	RC	■	□	□	□	■	■	□	■	Return	Ĉaretreveno
8E	SO	■	□	□	□	■	■	■	□	CTRL N	Alia kodo al la norma
0F	SI	□	□	□	□	■	■	■	■	CTRL O (litero)	Laŭ la norma kodo
90	DLE	■	□	□	■	□	□	□	□	CTRL P	Transmisieskapo
11	DC1	□	□	□	■	□	□	□	■	CTRL Q	Bendlegila startigo
12	DC2	□	□	□	□	■	□	■	□	CTRL R	Bendtruila startigo
93	DC3	■	□	□	■	□	□	■	■	CTRL S	Bendtruila haltigo
14	DC4	□	□	□	■	□	■	□	□	CTRL T	Bendtruila haltigo
95	NAK	■	□	□	■	□	■	□	■	CTRL U	Negativa kvitanco
96	SYN	■	□	□	■	□	■	■	□	CTRL V	Sinĥronigo.
17	ETB	□	□	□	■	□	■	■	■	CTRL W	Fino de transmisibloko
18	CAN	□	□	□	■	■	□	□	□	CTRL X	Ignoro
99	EM	■	□	□	■	■	□	□	■	CTRL Y	Mesaĝfino
9A	SUB	■	□	□	■	■	□	■	□	CTRL Z	Erarneniigo, anstataŭigo
1B	ESC	□	□	□	■	■	□	■	■	ESC	Eskapo
9C	FS	■	□	□	■	■	■	□	□	CTRL SHIFT L	Rikordara apartigilo
1D	GS	□	□	□	■	■	■	■	□	CTRL SHIFT M	Grupa apartigilo
1E	RS	□	□	□	■	■	■	■	□	CTRL SHIFT N	Rikorda apartigilo
9F	US	■	□	□	■	■	■	■	■	CTRL SHIFT O	Rikordparta apartigilo

16-uma kodo	Mnemoniko aŭ karakteo	Bendtruoj 8 7 6 5 4 3 2 1	Teletajpilaj klavoj	Rimarkigoj
A0	SP	■ □ ■ □ □ □ □ □	Spacigstango	Spaceto
21	!	□ □ ■ □ □ □ □ ■	SHIFT !	
22	''	□ □ ■ □ □ □ ■ □	SHIFT ''	Duobla apostrofo
A3	#	■ □ ■ □ □ □ ■ ■	SHIFT #	
24	$	□ □ ■ □ □ ■ □ □	SHIFT $	
A5	%	■ □ ■ □ □ ■ □ □	SHIFT %	
A6	&	■ □ ■ □ □ ■ ■ □	SHIFT &	
27	'	□ ■ □ □ □ ■ ■ ■	SHIFT '	
				Apostrofo
28	(	□ □ ■ □ ■ □ □ □	SHIFT (	
A9	)	■ □ ■ □ ■ □ □ ■	SHIFT)	
AA	*	■ □ ■ □ ■ □ ■ □	SHIFT *	
2B	+	□ □ ■ □ ■ □ ■ ■	SHIFT +	
AC	,	■ □ ■ □ ■ ■ □ □	,	Komo
2D	-	□ □ □ □ ■ ■ □ ■	-	Signo ''minus''
2E	.	□ □ ■ □ ■ ■ ■ □	.	Punkto
AF	/	■ □ ■ □ ■ ■ ■ ■	/	Signo ''divido''
30	0	□ □ ■ ■ □ □ □ □	0 (cifero 0)	
B1	1	■ □ ■ ■ □ □ ■ □	1	
B2	2	■ □ ■ ■ □ □ ■ □	2	
33	3	□ □ ■ ■ □ □ ■ ■	3	
B4	4	■ □ ■ ■ □ ■ □ □	4	
35	5	□ □ ■ ■ □ ■ □ ■	5	
36	6	□ □ ■ ■ □ ■ ■ □	6	
B7	7	■ □ ■ ■ □ ■ ■ ■	7	
B8	8	■ □ ■ ■ ■ □ □ □	8	
39	9	□ □ ■ ■ ■ □ □ ■	9	
3A	:	□ □ ■ ■ ■ □ ■ □	:	
BB	;	■ □ ■ ■ ■ □ ■ ■	;	
3C	<	□ □ ■ ■ ■ ■ □ □	SHIFT <	
BD	=	■ □ ■ ■ ■ ■ □ ■	SHIFT =	
BE	>	■ □ ■ ■ ■ ■ ■ □	SHIFT >	
3F	?	□ □ ■ ■ ■ ■ ■ ■	SHIFT ?	

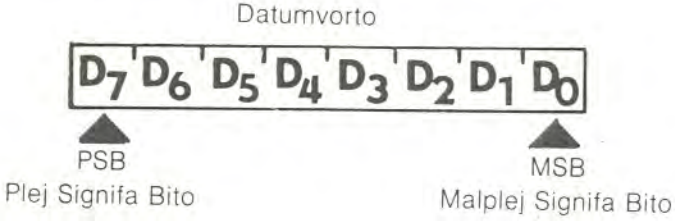
16-uma kodo	Mnemoniko aŭ karakteo	Bendtruoj 8 7 6 5 4 3 2 1	Teletajpilaj klavoj	Rimarkigoj
C0	X	■ ■ □ □ □ □ □ □	SHIFT	Komerca A
41	A	□ ■ □ □ □ □ □ ■	A	
42	B	□ ■ □ □ □ □ ■ □	B	
C3	C	■ ■ □ □ □ □ ■ ■	C	
44	D	□ ■ □ □ □ ■ □ □	D	
C5	E	■ ■ □ □ □ ■ □ ■	E	
C6	F	■ ■ □ □ □ ■ ■ □	F	
47	G	□ ■ □ □ □ ■ ■ ■	G	
48	H			
C9	I	□ ■ □ □ ■ □ □ □	H	
CA	J	■ ■ □ □ ■ □ □ ■	I	
4B	K	■ ■ □ □ ■ □ ■ □	J	
CC	L	□ ■ □ □ ■ □ ■ ■	K	
4D	M	■ ■ □ □ ■ ■ □ □	L	
AE	N	□ ■ □ □ ■ ■ □ ■	M	
CF	O	□ ■ □ □ ■ ■ ■ □	N	
		■ ■ □ □ ■ ■ ■ ■	O (litero 0)	
50	P	□ ■ □ ■ □ □ □ □	P	
D1	Q	■ ■ □ ■ □ □ □ ■	Q	
D2	R	■ ■ □ ■ □ □ ■ □	R	
53	S	□ ■ □ ■ □ □ ■ ■	S	
D4	T	■ ■ □ ■ □ ■ □ □	T	
55	U	□ ■ □ ■ □ ■ □ □	U	
56	V	□ ■ □ ■ □ ■ ■ □	V	
D7	W	■ ■ □ ■ □ ■ ■ ■	W	
D8	X	■ ■ □ ■ ■ □ □ □	X	
59	Y	□ ■ □ ■ ■ □ □ ■	Y	
5A	Z	□ ■ □ ■ ■ □ ■ □	Z	
DB	[	■ ■ □ ■ ■ □ ■ ■	SHIFT K	
5C	\	□ ■ □ ■ ■ ■ □ □	SHIFT L	
DD	]	■ ■ □ ■ ■ ■ □ ■	SHIFT M	
DE	↑	■ ■ □ ■ ■ ■ ■ □	SHIFT ↑	
5F	←	□ ■ □ ■ ■ ■ ■ ■	SHIFT ←	

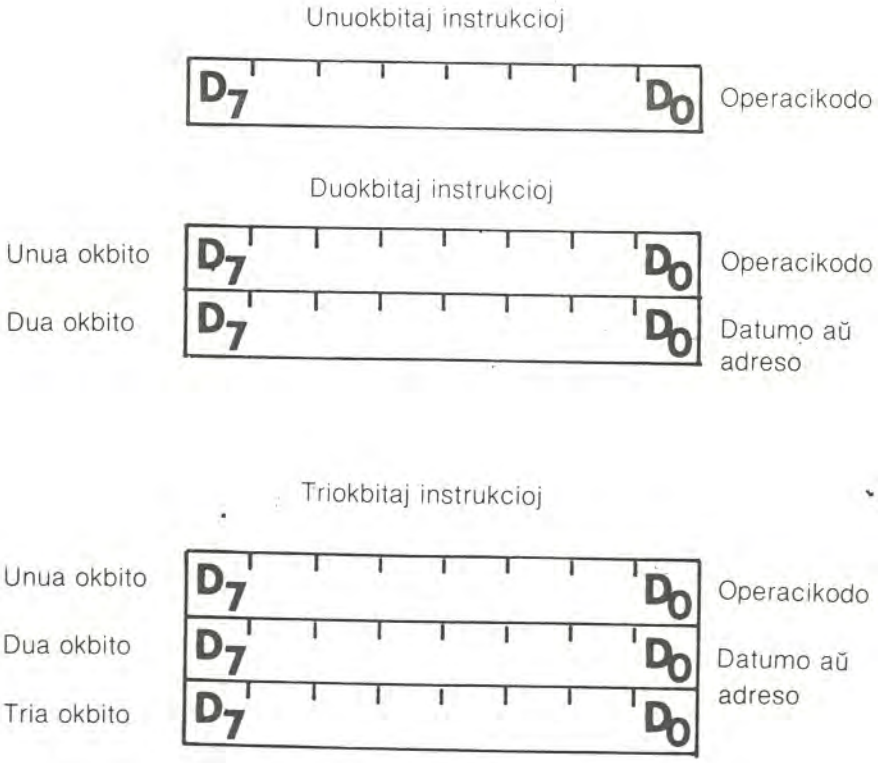
16-uma kodo	Mnemoniko aŭ karakteo	Bendtruoj 8 7 6 5 4 3 2 1	Teletajpilaj klavoj	Rimarkigoj
60		□ ■ ■ □ □ □ □ □		Presiĝas kiel
E1	a	■ ■ ■ □ □ □ □ ■		Presiĝas kiel A
E2	b	■ ■ ■ □ □ □ ■ □		Presiĝas kiel B
63	c	□ ■ ■ □ □ □ ■ ■		Presiĝas kiel C
E4	d	■ ■ ■ □ □ ■ □ □		Presiĝas kiel D
65	e	□ ■ ■ □ □ ■ □ ■		Presiĝas kiel E
66	f	□ ■ ■ □ □ ■ ■ □		Presiĝas kiel F
E7	g	■ ■ ■ □ □ ■ ■ ■		Presiĝas kiel G
E8	h	■ ■ ■ □ ■ □ □ □		Presiĝas kiel I
69	i	□ ■ ■ □ ■ □ □ ■		Presiĝas kiel J
6A	h	□ ■ ■ □ ■ □ ■ □		Presiĝas kiel K
EB	k	■ ■ ■ □ ■ □ ■ ■		Presiĝas kiel L
6C	l	□ ■ ■ □ ■ ■ □ □		Presiĝas kiel M
ED	m	■ ■ ■ □ ■ ■ □ ■		Presiĝas kiel N
EE	n	■ ■ ■ □ ■ ■ ■ □		Presiĝas kiel O
6F	o	□ ■ ■ □ ■ ■ ■ ■		
F0	p	■ ■ ■ ■ □ □ □ □		Presiĝas kiel P
71	q	□ ■ ■ ■ □ □ □ ■		Presiĝas kiel Q
72	r	□ ■ ■ ■ □ □ ■ □		Presiĝas kiel R
F3	s	■ ■ ■ ■ □ □ ■ ■		Presiĝas kiel S
74	t	□ ■ ■ ■ □ ■ □ □		Presiĝas kiel T
F5	u	■ ■ ■ ■ □ ■ □ □		Presiĝas kiel U
F6	v	■ ■ ■ ■ □ ■ ■ □		Presiĝas kiel V
77	w	□ ■ ■ ■ □ ■ ■ ■		Presiĝas kiel W
78	x	□ ■ ■ ■ ■ □ □ □		Presiĝas kiel X
F9	y	■ ■ ■ ■ ■ □ □ ■		Presiĝas kiel Y
FA	z	■ ■ ■ ■ ■ □ ■ □		Presiĝas kiel Z
7B	(	□ ■ ■ ■ ■ □ ■ ■		
FC	|	■ ■ ■ ■ ■ ■ □ □		
7D	)	□ ■ ■ ■ ■ ■ □ ■		
7E	~	□ ■ ■ ■ ■ ■ ■ □		
FF	DEL	■ ■ ■ ■ ■ ■ ■ ■	*Rub out*	Trupleno, erarneniigo.

Tab. 8: Internacia alfabeto n-ro 5. Rekomendo *V.3. du C.C.I.T.T.*
Internacia referenca tabelo.

b4	b3	b2	b1		b7	0	0	0	0	1	1	1	1
					b6	0	0	1	1	0	0	1	1
					b5	0	1	0	1	0	1	0	1
						0	1	2	3	4	5	6	7
0	0	0	0	0		NUL	TC (DLE)	SP	0	@	P	`	p
0	0	0	1	1		TC1 (SOH)	DC	!	1	A	Q	a	q
0	0	1	0	2		TC2 (STX)	DC2	"	2	B	R	b	r
0	0	1	1	3		TC3 (ETX)	DC3	#	3	C	S	c	s
0	1	0	0	4		TC4 (EOT)	DC4	¤	4	D	T	d	t
0	1	0	1	5		TC5 (ENQ)	TC8 (NAK)	%	5	E	U	e	u
0	1	1	0	6		TC6 (ACK)	TC9 (SYN)	&	6	F	V	f	v
0	1	1	1	7		BEL	TC10 (ETB)	'	7	G	W	g	w
1	0	0	0	8		FE0 (BS)	CAN	(	8	H	X	h	x
1	0	0	1	9		FE1 (HT)	EM	)	9	I	Y	i	y
1	0	1	0	10		FE2 (LF)	SUB	*	:	J	Z	j	z
1	0	1	1	11		FE3 (VT)	ESC	+	;	K	[	k	{
1	1	0	0	12		FE4 (FF)	IS4 (FS)	,	<	L	\	l	|
1	1	0	1	13		FE5 (CR)	IS3 (GS)	-	=	M	]	m	}
1	1	1	0	14		SO	IS2 (RS)	.	>	N	^	n	~
1	1	1	1	15		SI	IS1 (US)	/	?	O	_	o	DEL

Simboloj
TC: Transmisiregaj karaktroj
FE: Karaktroj por paĝa ordometo (aŭ prezento)
DC: Periferiajregaj karaktroj
IS: InformInterajoj.





Historio de la telefonaparatoj

Karlo Juhász (Hungario)*

La telefonaparato apartenas al la ĉiutaga vivo. En la artikolo mi prezentas la evoluon ekde la unua "telekrakilo" ĝis la nuntempe uzataj aparatoj.

Post naskiĝo de la telegrafo anonciĝis forta pretendo pri la teleparolo. Tiun plenumis *G. Bell*, kiu inventis la pratelefonon en 1876. Ĝi konsistas el elektromagneto, antaŭ ĝi situas membrano; vidu fig. 1. Se oni parolas, la membrano vibretas. La fenomeno induktas kurenton, kiu cirkvite fermiĝas tra simila aparato. Tie la kurento estigas magnetan flukson,kiu fine resonigas membranon. La aranĝaĵo estas nomata **elektromagneta mikrofono.**

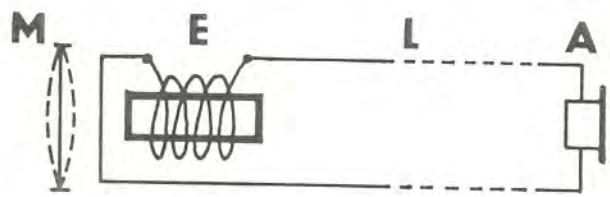


Fig. 1: La unua telefono: aparato de *Bell*.
M = membrano; E = elektromagneto; L = lineo; A = aŭdilo

La unua aparato funkciis modeste, tial *Bell* perfektigis ĝin. Modifante inventon de *Grav*, li tiam uzis la **rezistanco-mikrofonon**. Helpe de ĝi li telefone vokis sian kolegon. Tiu frazo estas konata kiel la unua frazo komunikita pertelefone: *Mr. Watson, come here. I want you!"* Jen la nova invento; fig. 2. Laŭ la parolintenso la membrano pli-malpli premadas la moveblan elek-

*Elektroinstrumentisto en la entrepreno B.H.G., Budapeŝto