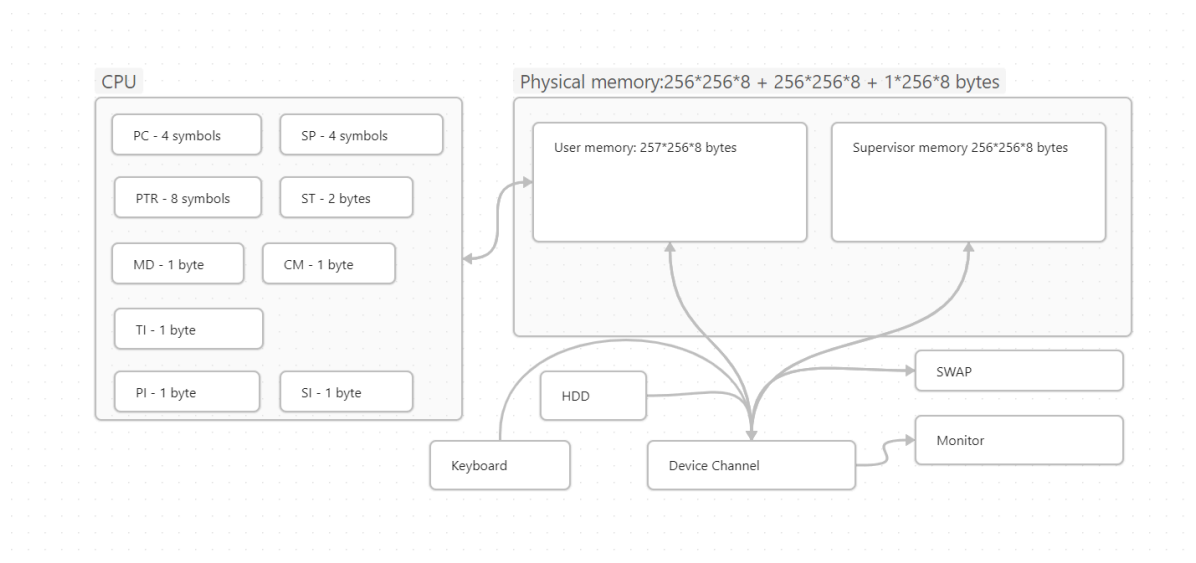


Reali mašina.



Pagrindinė atmintis: $256*256*8 + 256*256*8 + 1*256*8 = 1050624$ baitai.

Procesorius

Procesorius gali dirbti dviem režimais: naudotojo ir supervizoriaus. Naudotojo režime vykdomos virtualios mašinos komandos, o supervizoriaus režime – operacinės sistemos. Laiko tarpas, per kurį procesorius atlieka vieną komandą laikomas vienu laiko vienetu.

Registrai:

- PC — Program Counter, ilgis 4 simboliai: Kiekvieno procesoriaus takto pradžioje rodo į komandą kurį reikia atlikti per šitą taktą.
- PTR — Program Pointer, ilgis 8 simboliai: Rodo į programos puslapiavimo bloką.
- SP — Stack pointer, ilgis 4 simboliai: Kiekvieno procesoriaus takto pradžioje rodo į steko viršūnę.
- TI — Timer Interrupt, ilgis 1 baitas: Pradžioje turi reikšmę 9. Kiekvieno procesoriaus takto pradžioje tampa vienetų mažesnis. Kai pasiekia 0, vyksta pertraukimas. Po pertraukimo vėl gauna pradinę reikšmę.
- SI — System Interrupt, ilgis 1 baitas:

- 0 – nėra pertraukimo
- 1 – pertraukimas dėl dalybos iš nulio
- 2 – pertraukimas dėl steko perpildymas
- 3 – pertraukimas dėl nepasiekiamo VM adreso pasiekti bandymo
- 4 – pertraukimas dėl nepasiekiamo RM adreso pasiekti bandymo

- PI – Program Interrupt, ilgis 1 baitas:

- 0 – nėra pertraukimo
- 1 – pertraukimas skaitymui iš klaviatūros (fiksotas baitų skaičius)
- 2 – pertraukimas rašymui į ekraną (fiksotas baitų skaičius)
- 3 – pertraukimas skaitymui iš HDD (fiksotas baitų skaičius)
- 4 – pertraukimas skaitymui iš SWAP (fiksotas baitų skaičius)
- 5 – pertraukimas rašymui į SWAP (fiksotas baitų skaičius)
- 6 – pertraukimas skaitymui iš klaviatūros (kol nesutiksim reikalingą simbolį)
- 7 – pertraukimas rašymui į ekraną (kol nesutiksim reikalingą simbolį)
- 8 – pertraukimas skaitymui iš HDD (kol nesutiksim reikalingą simbolį)
- 9 – pertraukimas skaitymui iš SWAP (kol nesutiksim reikalingą simbolį)
- 10 – pertraukimas rašymui į SWAP (kol nesutiksim reikalingą simbolį)

- MD – Mode, ilgis 1 baitas: Ar mes esam supervizoriaus režime?

- = 0 jei mes ne esam supervizoriaus režime
- = 1 jei mes esam supervizoriaus režime

- CM – Computer mode, ilgis 1 baitas: konstanta, nurodanti, ar mes naudosim swap mechanizmą, ar mes bandysim taupiti atmintis.

- ST – Status, ilgis 2 baitai: Kiekvieno procesoriaus takto pradžioje turi požymius atsakančius į sekančius klausimus:

- Zero:
 - = 1 jei steko viršune nulis.
 - = 0 kitų atvejų
- Carry:
 - = 1 jei paskutines operacijos rezultatas neigiamas arba didesnis už musu didžiausią palaikomą skaičių (2^{32})
 - = 0 jei paskutines operacijos rezultatas mūsų skaičių apibrėžimo ribuose.

Atmintis.

Mes iš viso turėsime $256 \cdot 256 \cdot 8 + 256 \cdot 256 \cdot 8 + 1 \cdot 256 \cdot 8 = 1050624$ baitų atminties:

$256 \cdot 256 \cdot 8$ mes naudosime virtualiai mašinai, dar $1 \cdot 256 \cdot 8$ mes naudosime puslapiavimo blokams ir likusius $256 \cdot 256 \cdot 8$ baitus mes naudosime supervizoriniai atminčiai.

Be to atmintyje bus realizuotas swapinimo mechanizmas: bus specialus swap failas ir užkraunant naują programą, jei kompiuteriui neužteks atminties, dabar vykstantį programą bus išsaugota swap faile ir taip atmintis bus atlaisvinta naujai programai.

Puslapiavimo mechanizmas bus realizuotas sekančių būdų:

Specialus blokas turės adresus visų kitų VM mašinos blukų. Norint pasiekti tam tikrą adresą, kompiuteris žiūres į adreso bloko dalį ir pagal šią dalį prieis atitinkamą puslapiavimo bloko lastelę. Paskui kompiuteris prieis atitinkamo bloko pradžia pagal šią lastelę ir pridės prie šitos lastelės adreso antrą dalį*8.

Skaitant programos kodą iš HDD, mes galime matyti, kiek blokų naudos programą. Mes išskiriam tik tiek blokų, kiek bus reikalinga programai veikti. Jei programa norės padėti reikšmę į naują bloką, mes dinamiškai išskiriam atmintį šitam blokui. (remove paragraph)

Swappingo mechanizmas veiks sekančių būdų:

Kai mes išskiriam bloką programai, jei nėra laisvų blokų, mes padėsime viena iš kitų blokų į swap failą. Jei paskui reikės pasiekti šią bloką, tai mes dinamiškai ištrauksime jį iš swap failo.

Swap failas sistemai atrodo kaip adresavimo sistemos plėtinys. Adresai, kurių trečias iš kairės simbolis yra '1', priklauso swapo failui.

Adresavimas:

Realios mašinos adresai turės sekanti pavidalą:

Adresai nuo 00000000 iki 00020100 bus operacinės atminties adresai. Tarp jų adresai nuo 00000000 iki 0000ffff bus supervisorinės atminties adresai.

Adresai nuo 00100000 iki 001fffff bus SWAP failo adresai.

HDD atmintis.

Mūsų HDD bus implementuotas naudojant tekstinį failą. Atminties lastelės bus pirmi 8 baitai kiekvienos eilutės. Jei eilutėje mažiau nei 8 baitų, tai pirmi 8 baitai nuo paskutinio naujos eilutės simbolio.

Baitai po pirmų aštuonių bus naudojami požimiams.

Pvz:

____ ----// ← tuščias žodis

____ ----// komentaras

NAME1ver#NEW_PROGRAM // naujos programos požymis // naujos programos
pavadinimas: 'NAME1ver'

hello, w#DATA_SEGMENT

orld----

PSH 000c#CODE_SEGMENT

DS 00000

HALT ____ #NEW_PROGRAM_END // naujos programos pabaigos požymis

endlessL#NEW_PROGRAM // naujos programos požymis // naujos programos
pavadinimas: 'endlessL'

CMP ____ #CODE_SEGMENT

JMP 8000

HALT ____ #NEW_PROGRAM_END // naujos programos pabaigos požymis

addABprg#NEW_PROGRAM

Enter A:#DATA_SEGMENT

Enter B:

A + B =

PSH 0008#CODE_SEGMENT

DS 00000

PSH 000\

RDu00030

POP 0009

DS 00010

PSHa0009

RDu00040

POP 0009

POP 0009

PSHa0003

PSHa0004

ADDs ____

POP 0005

PSH 0008

DS 00020

DS 00050

HALT ____ #NEW_PROGRAM_END // naujos programos pabaigos požymis

SWAP failas.

SWAP failas tiesiog išpliečia operatyvų atmintį.

Kai mums neužtenka atminties, mes pernešam kai kuriuos blokus į SWAP failą, kai mums reikia tų blokų, mes pernešam kitus blokus į SWAP, o reikalingus ištraukiam iš SWAP failo.

Kanalų įrenginys.

Virtuali mašina nežino kad kanalų įrenginys egzistuoja.

Kanalų įrenginys turi registrus

SRC, 1 baitas — rodis šaltinio įrenginį.

SRCa, 5 baitai — rodis šaltinio įrenginio adresą su baitu tikslumu.

DES, 1 baitas — rodis gavėjo įrenginį.

DESa, 5 baitai — rodis gavėjo įrenginio adresą su baitu tikslumu.

SZ, 5 baitai — rodis perduodamų duomenų kiekį su baitu tikslumu.

MODE, 1 baitas — rodis kanalų įrenginio režimą, 2 režimai:

1 - perduoti fiksuota baitų skaičių (kuris nuroditas SZ registre).

2 - perdavinėti baitus kol nesutiksim baitą, kuris sutaps su paskutinių registro SZ baitų, arba kol baigs prieinama vieta.

Kanalai bus sunumeruoti

Gavėjo įrenginių numeracija (SRC registro reikšmių paaiškinimas):

Vartotojo atmintis — 1

Swapo erdvė — 2

Ekranas — 3

Supervizorinė — 4

Šaltinio įrenginių numeracija (DES registro reikšmių paaiškinimas):

Vartotojo atmintis – 1

Swapo erdve – 2

Klaviatura – 3

Supervizorine – 4

HDD – 5

Ne visos šaltinio ir gavejo kombinacijos veiks. Veikiančios kombinacijos (sintakse: šaltinis1, šaltinis2 → gavejas1, gavejas2):

2, 3, 4, 5 → 1

1 → 2, 3, 4

1, 2, 3, 5 → 4

4 → 1, 2, 3

Kanalų įrenginys tures viena operaciją: XCHG, virtualį mašiną nežinos apie šią operaciją.

Virtuali mašina.

VM Atmintis.

VM atmintis susideda iš 256 blokų po 256 žodžių. Vienas žodis sudarytas iš 8B, t. y. 64 Bit. Atmintis bus segmentuota ir susideda iš keturių segmentų:

- Duomenų segmentas, blokai: (00 ... 7f).
- Kodo segmentas, blokai: (80 ... ef).
- Steko segmentas, blokai: (f0 ... ff).

* rašant programą reikės nurodyti tik duomenų ir kodo segmentų pradžią.

Programos faile mes nurodam duomenų segmento pradžia ir programos kodo pradžia. Steko dydis yra vieta nuo f0 iki ff bloko pabaigos, t.y. 16 blokai, arba 16*256 žodžiai. Vartotojas galės naudoti VM atminties ląsteles nuo 0000 iki kodo segmento pradžios ląsteles (nuo 0000 iki 7fff).

Komandos:

Dauguma komandų turi sekančią sintaksę:

TTTTNNNN, kur TTTT - komandos pavadinimas, o NNNN - skaičius šešioliktainėje sistemoje. Jei komanda neturi argumentų, tai TTTT____, kur TTTT – komandos pavadinimas, o ____ - tušti baitai. Simboliai ‘_’ gali būti pakeisti į tarpus.

Aritmetinės komandos(laikoma, kad steke šešioliktainės skaičiai).

Aritmetinės komandos veiks tik su teigiamais sveikais skaičiais.

‘ADDs____’ – sudedam dvi paskutines steko reikšmes, rezultatas dedamas į steko viršūnę. Be to nustatom zero ir carry flagus pagal rezultatą.

‘SUBs____’ – atėmiam paskutinė steko reikšmė iš prieš paskutines, rezultatas dedamas į steko viršūnę. Be to nustatom zero ir carry flagus pagal rezultatą.

‘MULs____’ – dauginam dvi paskutines steko reikšmes, rezultatas dedamas į steko viršūnę. Be to nustatom zero ir carry flagus pagal rezultatą.

‘DIVs____’ – dalom paskutinė steko reikšmė iš prieš paskutines, liekaną dedama į steko viršūnę, o į priešpaskutinę steko reikšmę dedamas rezultatas. Be to nustatom zero flagą pagal liekaną.

Steko naudojimo komandos.

‘PSHaNNNN’ - (~“push”) dedame reikšmę adresų NNNN į steko viršūnę.

‘PSH NNNN’ - (~“push”) dedame reikšmę NNNN į steko viršūnę.

‘POP NNNN’ - ištraukiame iš steko viršaus reikšmę ir dedame į adresą NNNN.

‘TB1aNNNN’ - paversti steko viršūnę iš šešiolyktainio skaičiaus į binarinį ir padėti į pirmus keturius baitus adreso NNNN. (ne būtina)

‘TB2aNNNN’ - paversti steko viršūnę iš šešiolyktainio skaičiaus į binarinį ir padėti į paskutinius keturius baitus adreso NNNN. (ne būtina)

‘TH1aNNNN’ - paversti pirmus keturius baitus steko viršūnės iš binarinio skaičiaus į šešiolyktainį ir padėti į adresą NNNN. (ne būtina)

‘TH2aNNNN’ - paversti paskutinius keturius baitus steko viršūnės iš binarinio skaičiaus į šešiolyktainį ir padėti į adresą NNNN. (ne būtina)

Binarines operacijas (laikoma, kad steke šešiolyktaines skaičiai).

‘AND_____’ – pritaikom binarinę operaciją ‘and’ dviems paskutiniams steko reikšmėms. Rezultatas dedamas į steko viršūnę.

‘OR_____’ – pritaikom binarinę operaciją ‘or’ dviems paskutiniams steko reikšmėms. Rezultatas dedamas į steko viršūnę.

‘XOR_____’ – pritaikom binarinę operaciją ‘xor’ dviems paskutiniams steko reikšmėms. Rezultatas dedamas į steko viršūnę.

‘NOT_____’ – pritaikom binarinę operaciją ‘not’ paskutiniai steko reikšmės. Rezultatas dedamas į steko viršūnę.

Valdymo perdavimo komandos.

‘CMP_____’ Atimam paskutinę steko reikšmę iš priešpaskutinės, tačiau nekeičiam steką. Be to nustatom zero ir carry flagus pagal atimimo rezultatą.

‘JMP NNNN’ - perduoti valdymą adresui NNNN.

‘JMPBNNNN’ - perduoti valdymą adresui NNNN, jei carry flag = 1 (jei paskutinėje operacijoje rezultatas per didelis arba neigiamas skaičius).

‘JMPENNNN’ - perduoti valdymą, jei zero flag = 1 (jei paskutineje operacijoje rezultatas nulis).

‘JAE NNNN’ - perduoti valdymą adresui NNNN, jei carry flag = 0 (jei paskutineje operacijoje rezultatas apibrėžimo ribose).

‘JNE NNNN’ - perduoti valdymą adresui NNNN, jei zero flag = 0 (jei paskutineje operacijoje rezultatas ne nulis).

‘JBE NNNN’ - perduoti valdymą adresui NNNN, jei zero flag = 1 arba carry flag = 1 (jei paskutineje operacijoje rezultatas nedidesnis už nulį arba per didelis skaičius).

‘JMPANNNN’ - perduoti valdymą adresui NNNN, jei zero flag = 0 ir carry flag = 0 (jei paskutineje operacijoje rezultatas ne nulis ir apibrėžimo ribose).

Įrenginių naudojimo komandos.

‘DS NNNNN’ – išvesti į konsolę tiek baitų, kokia reikšmė turi steko viršūnė, pradedant nuo adreso NNNN, paskutine N yra baitų poslinkis nuo adreso pradžios (nuo 0 iki 15 (nuo 0 iki f)).

‘RD NNNNN’ – nuskaityti iš įvedimo srauto tiek baitų, kokia reikšmė turi steko viršūnė, ir įrašyti į atmintį pradedant nuo adreso NNNN, paskutine N yra baitų poslinkis nuo adreso pradžios (nuo 0 iki 15 (nuo 0 iki f)).

‘DSuNNNNN’ – išvedinėti į konsolę baitus, kol nesutiksim baitą, sutampantį su paskutinių steko viršūnės baitų, pradedant nuo adreso NNNN (bet sustoti išvedinėti jei pasieksim duomenų segmento pabaigą). Paskutine N yra baitų poslinkis nuo adreso pradžios (nuo 0 iki 15 (nuo 0 iki f)).

‘RDuNNNNN’ – nuskaityneti iš įvedimo srauto baitus, kol nesutiksim baitą, sutampantį su paskutinių steko viršūnės baitų, ir įrašyti į atmintį pradedant nuo adreso NNNN (bet sustoti rašyti duomenys jei pasieksim duomenų segmento pabaigą). Paskutine N yra baitų poslinkis nuo adreso pradžios (nuo 0 iki 15 (nuo 0 iki f)).

HALT_____ – programos pabaiga