1 Organizační úvod

Budeme se zabývat hlavně hardwarem + Operační systémy v letním semestru tím zbytkem. Volně navazuje i architektura počítačů v LS 2. ročníku. Chce se po nás umět souvislosti a termíny (občas budou české, ale vždy i s anglickou variantou). Budeme si lhát sofistikovaně.

2 Harvardská architektura

Podle univerzity

- procesor (CPU) – základní výpočetní jednotka
- kódová paměť (code memory) – paměť
- propojení – CPU přes něj čte data z paměti (měnit program (tedy to, co je v této paměti) neumí)
- datová paměť (data memory) – proměnné, lze do ni zapisovat (W) i z ní číst (R)
- input / output (I/O, tzv. periferie) – komunikace s okolím

Anglický (chudý) matematik Charles Babbage navrhl *Analytical engine* (1837, pohonem byl parní stroj, měl počítat daně), který měl tuto architekturu.

Ada Lovelace jeho manželka napsala manuál k Analytical engine (jako matematik to moc nepopsal). Zároveň vymyslela, že kromě čísel by šlo předávat i zakódovaná písmena, noty, atd.

3 Reprezentace (celých) čísel

3.1 přenos

- Můžeme reprezentovat analogově 1 jedním voltem, 2 dvěma, 10^6milionů.
- Můžeme použít i jinak naškálovanou hodnotu, 1 tisícinou voltu, 2 dvěma tisícinami 10^6 1000 voltů. Vodiče ale nejsou ideální vodiče, mění odpor / proud podle délky, tepla a elektromagnetického pole v okolí.
- Můžeme používat digitální (číslicový) přenos, třeba (někdy je to opačně): napětí nad nějakou hodnotou jako 1 a napětí pod tu hodnotou jako 0 (většinou je ta mez širší jen bod). Jedna taková hodnota je 1 bit (b). Přenášíme tzv. sériovým přenosem, každý bit chvíli (diagram zobrazující tenhle přenos je tzv. timing diagram).
- 1 a 0 jde i jinak, přes dva vodiče, podle směru jejich rozdílu (kladný vs. záporný rozdíl), což má navíc výhodu, že jsou šumem ovlivněny téměř stejně. Tzv. diferenciální přenos (např. USB).

Pozor
Napětí je relativní! (Musíme, krom diferenciálního přenosu, měřit oproti nějaké „nule“, tzv. zemi (ground).)

Pozor
Zdroj pracuje jen v uzavřeném obvodu!

Poznámka (Mocniny 2)
Hodí se naučit se mocniny 2: $2^4 = 64; 2^8 = 256; 2^10 = 1024; 2^12 = 4096; 2^16 = 65536; 2^20 \approx 10^6; 2^30 \approx 4,2 \cdot 10^9$

Definice 3.1 (Most significant bit (MSb))

Bit s nejvyšší hodnotou (nejvyšší mocninou 2).

Definice 3.2 (Least significant bit (LSb))

Bit s nejnižší hodnotou (nejnižší mocninou 2, typicky 2^0).

Definice 3.3 (Bitorder)

Pořadí, v jakém se bity posílají (MSb-first / LSb-first)

Definice 3.4 (Přenosová rychlost (transfer rate))

Rychlost, udává se v b / s (bps) nebo bauds (=symbols=včetně nedatových věcí jako mezery mezi „pakety“) / s.

Definice 3.5 (Idle stav linky)

Když se nic nepřenáší (lze vytvořit stavem vysoké impedance, nebo tím, že definujeme 1 / 0 jako idle stav a přenos je započat tzv. start condition = změna na 0 / 1 (jelikož je lepší start condition mít delší, tak se vysílá délkou 1 bit tzv. startbit)).

Definice 3.6 (stav Vysoké impedance = high Z)

Místo zapojení do 1 / do 0 vypojíme vysílací linku. Tento stav pak lze detekovat u přijímajícího).

Poznámka (Hodiny a synchronizace)
Start condition (začátek start condition je tzv. raising edge, na ní se právě synchronizuje) může dokonce synchronizovat čas.

Navíc hodiny nejsou přesné, takže po nějaké době budou rozsynchronizované. To lze vyřešit buď omezením délky přenosu. Většinou se však definuje délka, protože posílat end znak je těžké, tedy se dohodlo 8 bitů = 1 byte = 8b = 1B. (Avšak starší počítače měli

i jinou velikost 1B).

Po 1 přenosu však musí být pauza, aby se mohli zase synchronizovat hodiny, tedy se přenáší stopbity (začínají falling edge) (musí se dohodnout jejich počet, často 1).

To už začíná být moc zbytečných bitů, proto se přidává další signál – hodinový (střídající se 1 a 0, na hranách (ať už vzestupných nebo sestupných podle dohody, výjimečně ddr = double data rate = obojí) je střed bitu). Hodinový signál může vycházet buď od odesílatele, nebo z vnějšího zdroje pro oba.

Ten má ale problém, že se může (třeba různými délkami drátu) rozladit. Proto existuje 3 řešení: pravidelná synchronizace na hranách přenášeného signálu (clock recovery = obnovování hodin), aby se to nerozbíjelo na dlouhých shodných úsecích, existuje zakódování 8 bitů do 10 (nebo jiné, podle dohody), aby se nestalo, že všech 8 (nyní 10) bude jen 1 nebo jen 0 (např. USB, nebo RS232).

Poznámka (Duplexní přenos)
Zatím jsme řešili jen jednostrannou komunikaci (= simplexní přenos), ale v mnoha reálných situacích chceme tzv. duplexní přenos.

Nejjednodušší je halfduplex = přenáší se oběma směry po jedné lince.

Fullduplex jsou naopak 2 nezávislé jednosměrné (simplexní) linky (mohou mít společný ground) (např. RS232). Např. v RS232 je navíc linka pro tzv. out of bend signály (většinou buď 1 = pravda nebo 0 = nepravda, resp. tzv. inverzní logika značená overline) jako třeba warning nebo error (třeba 1 kabel pro informaci dochází baterka…).

Definice 3.7 (Řadič)

Dále je nutné definovat komunikační protokol, který může definovat různorodé pakety. Přijímání (tzn. třeba přechod z 7b na 8b byte, nebo LSb-first / MSb-first) má na starost tzv. řadič (= controller), paměti v řadiči se říká registr (= register).

Registry jsou podle funkce pojmenovány např. datový, buffer, konfigurační (pro konfiguraci např. rychlosti linky…), …

Pozor
Kromě čtení registrů potřebujeme i nastavit napájení myši a nejlépe ji i resetovat.

Poznámka
Dále jsme probírali dvojkovou a 16kovou soustavu + bitwise operace.

Poznámka
Včetně bitového posunu (shl = shift left = posouvá k MSb, ne doleva, …). A bitových rotací.

Poznámka
Dále jsme řešili reprezentaci záporných čísel (nejdříve 1 znaménkový bit, následně bitový doplněk a nakonec dvojkový doplněk).

Pozor
Čísla v Pythonu jsou divná. Proto jsme si ukazovali knihovnu NumPy, kde si můžeme požádat o konkrétně dlouhé číslo (abychom nepřišli o nulu na začátku).

Definice 3.8 (Master / slave)

Zařízením na jedné lince se říká master a slave, podle toho, kdo požaduje věci od koho. V počítači je často procesor master a slave budou tzv. zařízení (device).

Definice 3.9 (Multigroup linka = bus (sběrnice))

Procesor by musel mít zadrátováno mnoho linek, ale to je nepraktické, takže existují sběrnice (často se ale sběrnice říká i linkám). Na sběrnici je napojeno mnoho zařízení a v podstatě každé má tzv. adresu (masteři nemusí, ale také mohou). Každé zařízení musí být rozděleno na 2 části: r (receiver) a t (transmitter).

Aby nebyla sběrnice v nedefinovaném stavu, dělá se, že se na sběrnici připojí pull-up rezistor (napětí, tj. 1, připojené přes velký rezistor). Sběrnice se pak chová k vysílajícím signálům jako AND.

Definice 3.10 (I^2C = Inter Integrated Circuit)

Pomalá. Ideální na krátká propojení integrovaných obvodů. Řadič SDA (serial data). Má vestavěný hodinový signál SCL (serial clock) (v 0 se data mění, v 1 platí) (tiká pouze při komunikaci, generuje ho konkrétní master, který komunikuje).

Tzv. multimaster (může být více materů). Narozdíl třeba od USB sběrnice, která je singlemaster.

Přenos se startuje a ukončuje změnou stavu z jedné na nulu a z nuly na jedničku, když je hodinový signál 1 (neměl by se měnit).

Je devítibitový byte (8 data + 1 bit tzv. acknowledgement (= ACK = potvrzeno) (NACK = odmítnuto), potvrzení generuje přijímající). MSb-first. 1 = NACK, 0 = ACK, 1 se používá třeba pro ukončení přenosu, když má proměnnou délku, zároveň první potvrzuje, zda se zde zařízení, které je oslovováno, nachází. 100 kHz - 5 MHz.

Po sběrnici se přenáší pakety složené z management části (7bitová adresa + bit read = not write) a tzv. payloadu (dat, co chceme přenést).

Když slave nestíhá, může udržet hodinový signál v nule (tzv. clock stretching).

Například (ALS = Ambient Light Sensor (konkrétní od everlight))
(Detekuje intenzitu světla.) Má cca registr (ADC registr = analog digital converter = analog digitální převodník), který se při každém průletu fotonu zvýší o 1 (začátek = vynulování a konec si volíme sami při komunikaci).

Registr 2 bytový, 15 bitů číslo, 1 bit měří / neměří. Má adresu 0x29 danou od výrobce. Má pouze 1 read only (R/O) registr (počítací), má 1 write only (W/O) registr (příkazový), nemá žádný read write (R/W) registr.

Data se posílají LSB (sběrnice je MSb, zařízení LSB).

Například (256 bytová RAM (PCF8570))
V paměti potřebujeme tzv. paměťové adresy (= memory adress), které jsou v tzv. paměťovém adresovém prostoru (= rozsah adres v bitech, nemusí být plně využit). Občas se používá, že adresový prostor je o dost větší (8 bitů místo 16 bitů), protože uživatel časem může chtít upgradovat paměť a musel by přepisovat programy…

Základ paměti jsou tzv. Letche (4-6 tranzistorů pamatujících si 1bit), takovým pamětem se říká SRAM (Static Random Access Memory). 2 možnosti chápání random: 1.) Umíme číst odkudkoliv bez toho, abychom přečetli celou paměť. 2.) Umíme to pro všechny místa stejně rychle.

Důležitou vlastností RAM je, že nejsou RA. SRAM má sekvenční (od nejmenší adresy po největší) přístup (nejvíc jí „chutná“ čtení a zápis v adresách postupně, ale trochu zvládá i opačný směr, zato náhodný přístup jí nechutná). Hlavní vlastností tedy není RA, ale to, že z ní můžeme číst, můžeme do ní i zapisovat a je tzv. volatile (při odpojení napájení se data smažou).

SRAM (10-100 GB/s sekvenční čtení, přistupový čas (hledání náhodných adres, acces time) <1 ns) jsou drahé a velké (na stejnou paměť), takže existuje DRAM (podobné vlastnosti rychlosti přístupů, 1 tranzistor a 1 kondenzátor, takže jsou pomalejší (kondenzátor se vybíjí / nabíjí pomaleji), 1-10 GB/s, >1 ns).

DRAM navíc zapomínají obsah i při zapnutém napětí (řádově zapomínají v milisekundách), jelikož náboj se vybíjí z kondenzátoru do okolních kondenzátorů. Proto se provádí tzv. refresh (přečte se paměť a uloží se zase zpět), na což je v počítači sice součástka (aby to nedělal procesor), ale stejně z ní v té době nelze číst…

Definice 3.11 (kB, MB, GB, TB)

$$1TB = 2^{10}GB = 2^{20}MB = 2^{30}kB = 2^{40}B$$

Krom výrobců disku (marketingový tah).

Aby se to však odlišilo od fyzikálních jednotek, zavedla se jednotka KiB (KibiByte),

MiB (MibiByte), …, ale moc se nepoužívá.

Definice 3.12 (Word)

1 slovo (word) je délka jednoho přenosu.

Zařízení nazýváme n-bitové, podle délky slova.

Špatná definice 1 slovo = 16b.

Existuje i dword = dvojslovo a qword = čtyřslovo.

Paměť obsahuje tzv. bus interface (část, která komunikuje se sběrnicí) a v ní je např. address register, ze kterého čte logika tahající a ukládající data z a v paměti.

4 Zpět k základům

5 Procesor

Definice 5.1 (Instrukce)

Základní příkaz pro procesor. Tzv. instrukční sada procesoru je množina instrukcí, které procesor umí.

„Seznam“ instrukcí (program) je strojový kód (machine code).

Identifikátoru instrukce se říká OP code (operační kód), zbytek jsou argumenty / parametry (vzhledem k velikosti argumentům je omezena velikost paměti).

Definice 5.2 (Procesorové registry)

Procesor má také registry, aby mohl ukládat svůj stav. Například existuje registr Instruction pointer (= IP = někdy Program counter = PC), který ukládá, která instrukce se vykonává. (Více bytové instrukce reprezentuji ukazatelem na první byte.) Velikost PC bude taková, aby obsáhla code memory. (Proto je velikost code memory omezena podle procesoru.)

Definice 5.3 (Programování)

Když programujeme, tak v code memory je uložené např. Visual Studio a v normální paměti je uložen program v textu. V tuto chvíli program nemůžeme na Harvardské architektuře tento program spustit. Proto máme tzv. překladač, který přeloží kód z Pythonu do strojového kódu. Stále se to však v H. architektuře nemůže běžet, ale procesory mají v sobě „magii“, která umí nakopírovat data z paměti do kódové paměti. Pak už daný program může běžet…

Ve skutečnosti Python běží v interpretru (interpreter), který umí číst kód v Pythonu

a rovnou se podle něho chovat. Dělá se to kvůli tomu, že napsat interpreter je jednodušší než napsat překladač. Zase na druhou stranu, přeložený kód je daleko rychlejší.

Little endian (LE) a Big endian je inspirované příběhem s vajíčkem z Gulliverových cest a značí „LSB-first“ a „MSB-first“. Instrukce musí mít danou endianitu, proto každý procesor má dáno, zda je little endian (dnes převažují) nebo big endian (dost často třeba mobilní procesory). Endianitu musíme znát kvůli dalším zařízením připojeným k našemu procesoru.

Poznámka
Harvardskou architekturu (musíme code memory mít non-volatile) už nebudeme mít rádi.

6 Von Neumannova architektura

Definice 6.1

Procesor je spojen pouze s jednou pamětí, kde na začátku je kód, který se vykonává.

Poznámka (Historie)
Počítače začali jako UNIVAC (1951, obrovský drahý, počítal mzdy). Následně pro použití studentům (potřebovali menší verzi) existoval altair 8800 (1974, začátek Basicu). Následoval Apple I (1976) pro širokou veřejnost (1976, Steeve Jobs to prodal, cena 666 dolarů), o rok později Apple II za dvojnásobnou cenu (prodávali to s VisiCalc = předchůdce excelu = počítání daňových přiznání). Následuje obrovské množství počítačů jako např. Atari 2600 (1977), Atari 1979 (1979), …

Procesor 6503 (pro příklad): 8bitový (slovo = 8 b), Von Neumannovská architektura, 16b adresový prostor, little endian. Úspěchu tohoto procesoru si všimla IBM → intel: 16bitový, 20b adresový prostor, little endian (později se velmi ukázala výhoda velikosti adresového prostoru). AMD a následně Intel pak vytvořili 32b procesory, které mají i instrukce pro staré procesory, aby byly zpětně kompatibilní.

Definice 6.2 (Seznam běžných instrukcí)

Instrukce „nedělej nic“. Jump / branch instrukce (včetně podmíněného skoku). STA (store acumulator), LDA. Bitwise operace. Shiftování ASL.

Strojový kód se dá zapisovat v tzv. assembleru (NOP, JMP). Překladači z assembleru se říká assembler ;)

Konstanta = immediate → immediate instrukce pracují s konstantami (značené často #).

Definice 6.3 (Speciální / jiné registry)

Aritmetické instrukce většinou moc neumí pracovat s pamětí a existují proto registry, ze kterých se pak čísla, se kterými se má provést operace, berou. Navíc proto existují instrukce LOAD a STORE, které načtou číslo z paměti do registru a uloží ho zpět.

Speciální jsou např. IP.

Definice 6.4 (Příznak)

Tzv. flag. je jednobitová hodnota.

Příznakový registr: bytový registr, kde každý bit je jiný příznak. Zároveň procesory často mají příkazy set* a clear*, kde * je zkratka příznaku, které nastavují daný příznak na 1 a 0.

Běžné příznaky: zero (byl výsledek poslední operace nula? 1 = byl, 0 = nebyl), negative (byl výsledek poslední operace záporný? 1 = byl, 0 = nebyl, nepozná ale, zda byly čísla signed, takže to dělá vždycky), carry (přenos, používá se pro aritmetické operace s vícebitovými čísly, než s kterými umíme pracovat, sčítání s přenosem se nazývá add with carry: ADC, zároveň existuje odečítání (s vypůjčením = borrow): SBB (často se zde „zneužívá“ carry jako borrow), nebo SBC (používá se negace carry))

Poznámka (Proč je python pomalý)
Čísla ukládá jako minimálně 32 bitová (a při každé operaci se musí znormalizovat na stejnou velikost, navíc musí být zmenšený rozsah, aby číslo nepřeteklo / nepodteklo), každá proměnná je pointer, reálná data proměnné obsahují vždy minimálně type a další 4 byty, kolik ukazatelů na danou proměnnou vede (pro garbage collector). Navíc se musí interpretovat.

Tedy je cca 100krát pomalejší.

Poznámka (Aritmetic overflow (a underflow))
Na rozdíl od Pythonu se v dalších jazycích může stát, že nám číslo tzv. přeteče (ořeže se nejvyšší bit, protože se už nevejde do paměti (např. 9. bit u 8-bitového čísla)).

Poznámka (Násobení)
Dost počítačů (to se netýká stolních počítačů a notebooků, i mobily dost často mají násobení) nemá instrukci násobení / dělení.

Násobí / dělí se pomocí sčítání / odčítání, bitových posunů a podmíněných skoků.

U mocnin dvojky si můžeme pomoci tak, že to rovnou ručně přepíšeme do posunů. (I když dnes už jsou překladače dostatečně „chytré“, aby to nahradily samy.)

Pozor
Toto lze dělat jen pro bezznaménková čísla. Proto existuje ještě tzv. aritmetický posun doprava (SAR), který (při posunu) rozkopírovává nejvyšší bit. Ale stejně špatně funguje zaokrouhlování.

V Pythonu se používá pouze SHL a SAR. V javě jsou naopak SHL (<<), SAR (>>) i SAR (>>>). V C# se dokonce tyto operátory chovají podle typu čísel.

7 Reálná čísla

Definice 7.1 (Ukládání)

Například se uloží jako číslo před desetinnou čárkou a číslo (s pevnou délkou) za desetinnou čárkou (mezinárodně tečkou).

Poznámka
Dále jsme celou přednášku probírali floating point čísla (reprezentují se jako mantisa a exponent, mantisa vždy začíná jedničkou, takže ta se neukládá, porovnávat se musí pomocí nějakého epsilon a vzdálenosti, nula je to nejmenší číslo, obsahují stav NaN, …)

Definice 7.2 (ROM paměť)

Na rozdíl od RAM paměti na ni nelze zapisovat (nebo pokud lze, tak jen jednou, program se tam uloží tak, že se přepálí dané diody).

Definice 7.3 (EEPROM)

Electrically EPROM = už je plně Read / Write narozdíl od EPROM, kterou lze měnit jen ultrafialovým světlem (poznáme podle pásky, kterou bývá přelepená).

= NVRAM = Non-volatile RAM (VR jako write read)…

Spolu s FLASH mají výhodu, že jsou relativně rychlé a lze z nich libovolněkrát číst. Nelze do nich však libovolněkrát zapisovat.

Definice 7.4 (Permanentní datové úložiště)

Nebo také storage device je zařízení, které musí být non-volatile, protože si do něj chceme ukládat věci, které zůstanou po vypadnutí proudu.

Například Hard Disk Drive (HDD), kde se ukládají data magneticky na kruhové stopy, které jsou výsečemi rozděleny na sektory (dříve 512B, dnes 4kB). Navíc ještě často mají několik ploten (k nim ke každé čtecí hlavu, které se však po mezi stopami pohybují současně), které se při startu roztočí a pak se prostě točí. Pokud se nepoškodí, tak mají libovolně zápisů a čtení, mají největší kapacitu a data se z nich neztrácí…

… ale jsou pomalé. Hlavně, když bychom chtěli číst náhodně, musí se čekat na otáčky i posuny hlavy. Navíc kvůli rozdělení sektorů výsečí se disk otáčí „rychleji“ na kraji, tedy se snažíme ukládat hlavně na okraj (menší indexy, je indexován zvenku dovnitř).

Podobně fungují CDčka, DVD a BlueRay. Místo magnetu fungují na principu odrazu (povrch se připraví buď v továrně, pak tam data spíše vydrží dlouho, nebo se vypálí, ale pak má materiál tendenci se vracet do vymazaného stavu). Na rozdíl od hard disku má sektory stejně dlouhé (nejsou to výseče), data mají místo kruhů ve spirále a točí se daleko pomaleji.

Adresování CHS (HDD, plotny + stopy) a LBA (CD, lineární adresní prostor)? Řadič si zpravidla bude ukládat vždy 1 sektor, tedy bude mít registr na paměť, data buffer, data registr a info registr (na parametry disku: kapacita, velikost sektoru, …). Adresuje se LBA, řadič disku si to překládá na CHS (nejdřív plotny, pak až stopy, aby se zapisovalo spíše na vnější povrch).

SSD (solid state disk) je např. flash paměť s protokolem jako u HDD. Je rychlejší, ale menší a má menší počet zápisů.

Definice 7.5 (GPIO)

General purpose input output = plně přístupné elektrické kontakty (piny na arduinu / RaspberryPi)

Definice 7.6 (Soubro)

Skupina dat (bytů). Disk o ní něm nemá ponětí, proto na něm musí být uložena i metadata (data o datech: kde jsou uloženy, co za data to je, …), protože přepisováním dat se disk tzv. fragmentuje (data nejsou kontinuálně vedle sebe).

Definice 7.7 (Souborový systém (file system))

Definuje, jak jsou uložena meta data + formát uložení dat.

Aby každý nemusel implementovat čtení disku, každý programovací jazyk si umí vytáhnout tyto funkce z tzv. operačního systému.

Definice 7.8 (Otevírání souborů)

Pokud soubor otevřeme ve ’r’, tak se zavolá funkce systému na čtení souboru a zapne se (např. Python) zpracování rovnou do znaků. Pro čtení bytů se musí nastavit ’rb’. Obdobně pro zápis.

Čtením / zápisem se posouvá offset o přečtenou / zapsanou část. Existuje i příkaz (klasicky seek, tzv. seekovat), kterým lze nastavit offset *od začátku souboru!*

Definice 7.9 (Obrázky)

Ukládá se bgra.

Definice 7.10 (Formát, header)

Soubory potřebují metadata a navíc se data můžou ukládat různě, proto existují formáty souborů. Metadata jsou většinou na začátku a pak se jim říká header. Dokonce většinou obsahují konstantu (magic constant, signature), tj. první byt je v tomto formátu vždy shodný, např. BMP má BM.

8 Reprezentace textu

Definice 8.1 (Kódování)

Bijekce mezi znaky (tj. obrázky = grafémy) a kódy (číselná reprezentace). Celý text je v 1 kódováním. Znaky se ukládají za sebou v pořadí, v jakém se čtou.

Dělí se na kódování s pevnou délkou (1 bytové, 2 bytové, …) a kódování s proměnnou délkou kódu (různé znaky mohou mít různé délky kódu).

Vniklo mnoho kódování, ale naštěstí vzniklo ASCII (American Standard Code for Information Interchange). Má několik nevýhod (vzniklo v 70 letech): je pouze 7bitové (protože tehdy byly i 7b přenosy), implementuje jenom americké znaky. Naopak má i výhody: abeceda (jak malá, tak velká) i číslice jsou uloženy za sebou (pozor, '0' $\neq$ 0).

Jenom české kódování jsou 2 velmi používaná: ISO8859-2 – Linux ([ISO] latin2, na Windowsech je značeno: 28592), (852 – DOS (latin2)) a win1250 – windows. Což je dost špatné, protože textové soubory nepoužívají metadata.

Definice 8.2 (Unicode)

Definuje pouze přechod od znaku ke kódu, ale neříká, jak se bude kód ukládat (blbci ;)). Tedy existuje několik různých překladů kód -> uložení, například: UTF-32le (častější), UTF-32be, UTF-8, ….