MARVIN

'Marvin' est un Acronyme Récursif Vaguement INtrospectif

Naïm Favier

21 janvier 2020

Simulateur

- deux interfaces :

Simulateur

- deux interfaces :
  - « dialogue » : lit les variables d'entrée au clavier et affiche les variables de sortie

Simulateur

- deux interfaces :
  - « dialogue » : lit les variables d'entrée au clavier et affiche les variables de sortie
  - « horloge » : gère l'affichage de l'horloge et la synchronisation

Simulateur

- deux interfaces :
  - « dialogue » : lit les variables d'entrée au clavier et affiche les variables de sortie
  - « horloge » : gère l'affichage de l'horloge et la synchronisation
- on ne calcule que ce qui est nécessaire :

Simulateur

- deux interfaces :
  - « dialogue » : lit les variables d'entrée au clavier et affiche les variables de sortie
  - « horloge » : gère l'affichage de l'horloge et la synchronisation
- on ne calcule que ce qui est nécessaire :
  - variables de sortie

Simulateur

- deux interfaces :
  - « dialogue » : lit les variables d'entrée au clavier et affiche les variables de sortie
  - « horloge » : gère l'affichage de l'horloge et la synchronisation
- on ne calcule que ce qui est nécessaire :
  - variables de sortie
  - registres (`... = REG x`)

Simulateur

- deux interfaces :
  - « dialogue » : lit les variables d'entrée au clavier et affiche les variables de sortie
  - « horloge » : gère l'affichage de l'horloge et la synchronisation
- on ne calcule que ce qui est nécessaire :
  - variables de sortie
  - registres (`... = REG x`)
  - accès à la RAM (`x = RAM ...`)

Simulateur

- deux interfaces :
  - « dialogue » : lit les variables d'entrée au clavier et affiche les variables de sortie
  - « horloge » : gère l'affichage de l'horloge et la synchronisation
- on ne calcule que ce qui est nécessaire :
  - variables de sortie
  - registres (`... = REG x`)
  - accès à la RAM (`x = RAM ...`)
- les variables sont représentées par des entiers (accès en $O(1)$)

Simulateur

- deux interfaces :
  - « dialogue » : lit les variables d'entrée au clavier et affiche les variables de sortie
  - « horloge » : gère l'affichage de l'horloge et la synchronisation
- on ne calcule que ce qui est nécessaire :
  - variables de sortie
  - registres (`... = REG x`)
  - accès à la RAM (`x = RAM ...`)
- les variables sont représentées par des entiers (accès en $O(1)$)
- la RAM (resp. ROM) est initialisée à partir de l'image `ram.img` (resp. `rom.img`)

Microprocesseur

- architecture 42 bits

Microprocesseur

- architecture 42 bits
- registres : `a`, `b`, `c`, `d`, `pc`

Microprocesseur

- architecture 42 bits
- registres : `a`, `b`, `c`, `d`, `pc`
- instructions à taille fixe : un *opcode* et deux opérandes (pas forcément utilisées)

Microprocesseur

- architecture 42 bits
- registres : `a`, `b`, `c`, `d`, `pc`
- instructions à taille fixe : un *opcode* et deux opérandes (pas forcément utilisées)
- modes d'adressage : immédiat, registre, mémoire (2 bits de poids faible)

Microprocesseur

- architecture 42 bits
- registres : `a`, `b`, `c`, `d`, `pc`
- instructions à taille fixe : un *opcode* et deux opérandes (pas forcément utilisées)
- modes d'adressage : immédiat, registre, mémoire (2 bits de poids faible)
- instructions :

Microprocesseur

- architecture 42 bits
- registres : `a`, `b`, `c`, `d`, `pc`
- instructions à taille fixe : un *opcode* et deux opérandes (pas forcément utilisées)
- modes d'adressage : immédiat, registre, mémoire (2 bits de poids faible)
- instructions :
  - déplacement : `mov`

Microprocesseur

- architecture 42 bits
- registres : `a`, `b`, `c`, `d`, `pc`
- instructions à taille fixe : un *opcode* et deux opérandes (pas forcément utilisées)
- modes d'adressage : immédiat, registre, mémoire (2 bits de poids faible)
- instructions :
  - déplacement : `mov`
  - arithmétique : `add`, `sub`, `mul`, `div`, `mod`

Microprocesseur

- architecture 42 bits
- registres : `a`, `b`, `c`, `d`, `pc`
- instructions à taille fixe : un *opcode* et deux opérandes (pas forcément utilisées)
- modes d'adressage : immédiat, registre, mémoire (2 bits de poids faible)
- instructions :
  - déplacement : `mov`
  - arithmétique : `add`, `sub`, `mul`, `div`, `mod`
  - logique : `not`, `and`, `or`, `xor`, `test`

Microprocesseur

- architecture 42 bits
- registres : `a`, `b`, `c`, `d`, `pc`
- instructions à taille fixe : un *opcode* et deux opérandes (pas forcément utilisées)
- modes d'adressage : immédiat, registre, mémoire (2 bits de poids faible)
- instructions :
  - déplacement : `mov`
  - arithmétique : `add`, `sub`, `mul`, `div`, `mod`
  - logique : `not`, `and`, `or`, `xor`, `test`
  - sauts : `jmp`, `jz`, `jnz`, `jl`, `jg`, `jle`, `jge` (registre `a`)

Microprocesseur

- architecture 42 bits
- registres : `a`, `b`, `c`, `d`, `pc`
- instructions à taille fixe : un *opcode* et deux opérandes (pas forcément utilisées)
- modes d'adressage : immédiat, registre, mémoire (2 bits de poids faible)
- instructions :
  - déplacement : `mov`
  - arithmétique : `add`, `sub`, `mul`, `div`, `mod`
  - logique : `not`, `and`, `or`, `xor`, `test`
  - sauts : `jmp`, `jz`, `jnz`, `jl`, `jg`, `jle`, `jge` (registre `a`)
- signaux de contrôle encodés dans les bits de l'*opcode*

Microprocesseur

- architecture 42 bits
- registres : `a`, `b`, `c`, `d`, `pc`
- instructions à taille fixe : un *opcode* et deux opérandes (pas forcément utilisées)
- modes d'adressage : immédiat, registre, mémoire (2 bits de poids faible)
- instructions :
  - déplacement : `mov`
  - arithmétique : `add`, `sub`, `mul`, `div`, `mod`
  - logique : `not`, `and`, `or`, `xor`, `test`
  - sauts : `jmp`, `jz`, `jnz`, `jl`, `jg`, `jle`, `jge` (registre `a`)
- signaux de contrôle encodés dans les bits de l'*opcode*
  - e.g. `sub = signal_addsub + signal_sub + signal_mov`

Assembleur

- génère une image RAM contenant le programme exécutable à l'adresse 0

Assembleur

- génère une image RAM contenant le programme exécutable à l'adresse 0
- étiquettes : `foo:`

Assembleur

- génère une image RAM contenant le programme exécutable à l'adresse 0
- étiquettes : `foo:`
- instructions : `ins [op1 [op2]]`

Assembleur

- génère une image RAM contenant le programme exécutable à l'adresse 0
- étiquettes : `foo:`
- instructions : `ins [op1 [op2]]`
- opérandes :

Assembleur

- génère une image RAM contenant le programme exécutable à l'adresse 0
- étiquettes : `foo:`
- instructions : `ins [op1 [op2]]`
- opérandes :
  - registres : `%a`

Assembleur

- génère une image RAM contenant le programme exécutable à l'adresse 0
- étiquettes : `foo:`
- instructions : `ins [op1 [op2]]`
- opérandes :
  - registres : `%a`
  - références mémoire : `*foo`, `*42`

Assembleur

- génère une image RAM contenant le programme exécutable à l'adresse 0
- étiquettes : `foo:`
- instructions : `ins [op1 [op2]]`
- opérandes :
  - registres : `%a`
  - références mémoire : `*foo`, `*42`
  - littéraux : `foo`, `42`

Assembleur

- génère une image RAM contenant le programme exécutable à l'adresse 0
- étiquettes : `foo:`
- instructions : `ins [op1 [op2]]`
- opérandes :
  - registres : `%a`
  - références mémoire : `*foo`, `*42`
  - littéraux : `foo`, `42`
- variables : `x = y`

Horloge

- emplacements mémoire réservés pour la communication avec le simulateur :

Horloge

- emplacements mémoire réservés pour la communication avec le simulateur :
  - 1024-1029 : secondes, minutes, heures, jours, mois, années

Horloge

- emplacements mémoire réservés pour la communication avec le simulateur :
  - 1024-1029 : secondes, minutes, heures, jours, mois, années
  - 1030 : mode synchrone ?

Horloge

- emplacements mémoire réservés pour la communication avec le simulateur :
  - 1024-1029 : secondes, minutes, heures, jours, mois, années
  - 1030 : mode synchrone ?
  - 1031 : secondes écoulées

Horloge

- emplacements mémoire réservés pour la communication avec le simulateur :
  - 1024-1029 : secondes, minutes, heures, jours, mois, années
  - 1030 : mode synchrone ?
  - 1031 : secondes écoulées
- boucle :

Horloge

- emplacements mémoire réservés pour la communication avec le simulateur :
  - 1024-1029 : secondes, minutes, heures, jours, mois, années
  - 1030 : mode synchrone ?
  - 1031 : secondes écoulées
- boucle :
  - on incrémente les secondes

Horloge

- emplacements mémoire réservés pour la communication avec le simulateur :
  - 1024-1029 : secondes, minutes, heures, jours, mois, années
  - 1030 : mode synchrone ?
  - 1031 : secondes écoulées
- boucle :
  - on incrémente les secondes
  - si `secondes >= 60` :

Horloge

- emplacements mémoire réservés pour la communication avec le simulateur :
  - 1024-1029 : secondes, minutes, heures, jours, mois, années
  - 1030 : mode synchrone ?
  - 1031 : secondes écoulées
- boucle :
  - on incrémente les secondes
  - si `secondes >= 60` :
    - on remet les secondes à 0

Horloge

- emplacements mémoire réservés pour la communication avec le simulateur :
  - 1024-1029 : secondes, minutes, heures, jours, mois, années
  - 1030 : mode synchrone ?
  - 1031 : secondes écoulées
- boucle :
  - on incrémente les secondes
  - si `secondes >= 60` :
    - on remet les secondes à 0
    - on incrémente les minutes

Horloge

- emplacements mémoire réservés pour la communication avec le simulateur :
  - 1024-1029 : secondes, minutes, heures, jours, mois, années
  - 1030 : mode synchrone ?
  - 1031 : secondes écoulées
- boucle :
  - on incrémente les secondes
  - si `secondes >= 60` :
    - on remet les secondes à 0
    - on incrémente les minutes
    - ...

Questions ?

`https://git.monade.li/marvin`