Introduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion

ELEC288 : Architectures et systèmes à micro-processeurs

La famille Zilog Z80

Guillaume Desmottes

29 mai 2006



- 1974 : Fondation de ZiLOG par Frederico Faggin
- 1976 : Lancement de la famille des Z80
- Un des processeurs 8 bits les plus populaires

(D) (B) (돌) (돌) 돌 키익(P

Introduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion

Utilisation

- ordinateurs 8 bits: TRS-80, ZX80, ZX Spectrum, ...
- Commodore 64 et 128
- Console de jeux vidéos : GameBoy, Master System, Game Gear
- Calculatrices Texas Instrument
- Très utilisé pour μ C



- Processeur 8 bits
- Compatibilité binaire avec le 8080 d'Intel
- Philosophie CISC : instruction de 1 à 4 bytes
- Fréquence : de 2.5 à 20 Mhz

(D) (B) (돌) (돌) 돌 키익(P

Introduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion

Architecture

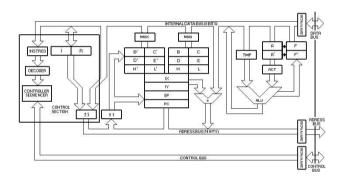


Figure: Architecture du Z80

(日) (원) (분) (분) 분 되었다.

ntroduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion

Améliorations p/r au 8080

- Jeux d'instructions étendus (80): manipulation de bit et de strings, transfert de blocs, recherche de byte
- Deux registres d'index ainsi que les instructions ad-hoc
- Deux bancs de registres dissociés pouvant être rapidement switchés
- Contrôleur DRAM
- Alimentation unique 5V
- Meilleur prix



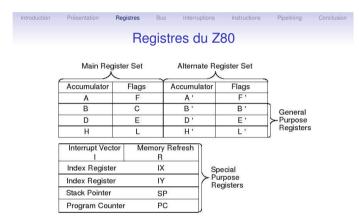


Figure: Registres du Z80

<□> <**□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**> < **□**

Introduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion

Registres compatibles 8080

- Accumulateur A (8 bits) et Flag F associé (carry, signe, 0, ...)
- 6 registres de 8 bits pouvant être utilisés par paire : BC, DE, HL
- Program Counter 16 bits
- Stack Pointer SP de 16 bits



Introduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion

Registres introduits par le Z80

- Deux registres de 16 bits pour l'adressage indexé : IX et IY
- Registre *Interrupt* I sur 8 bits : forme 8 bits de poids fort de l'adresse du vecteur d'interruption
- Registre Refresh R sur 8 bits
- Un deuxième ensemble de registres A'F', B'C', D'E', H'L'

←□→ ←□→ ←□→ ←□→ □ ● ◆9

Introduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion

MUX

- Permet de switcher très rapidement AF, BC, DE et HL par A'F', B'C', D'E' et H'L'
 - EX AF, A'F': pour les routines vraiment très simples
 - EXX : pour échanger les autres registres
- Surtout utilisé pour les routines de gestion d'interruption
 - Le programme utilise les registres principaux
 - Les routines de gestion utilisent le set EXX
 - Permet d'éviter de devoir sauver / restaurer le contexte en mémoire à chaque fois
 - → gros gain en performances



Introduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion Bus

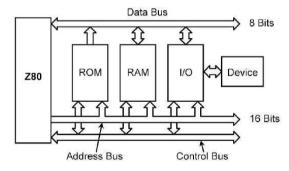


Figure: Bus du Z80

(D) (B) (E) (E) E 990

Introduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion Bus

- Bus de données : 8 bits
- → les données sont transférées 1 byte / cycle
- Bus d'adresses : 16 bits
- \rightarrow Espace d'adressage : 2^{16} = 65536 bytes = 64 KB
- Bus de contrôle
 - System Control: M1, MREQ, IORQ, RD, WR, RFSH
 CPU Control: HALT, WAIT, INT, NMI, RESET

 - CPU Bus Control: BUSRQ, BUSACK

(□) (□) (□) (Ξ) (Ξ) (Ξ) (Ξ) (Θ

Figure: Z80 I/O Pin Configuration

(ロ) (**日**) (言) (言) (言) (を

Introduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion

Type d'interruptions

- Deux signaux d'interruptions
 - INT : masquable par logiciel
 - NMI : non-masquable
- IFF1: indique si on active ou non les interruptions masquables
- IFF2 : sauvegarde l'état de IFF1 lors des traitements des interruptions non-masquables
- Lorsque qu'on accepte une interruption, IFF1 est automatiquement désactivé
- Instructions spécifiques : EI, DI, RETI, RETN



Introduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion Gestion des interruptions

- Non masquables : saute en 0066
- 3 types de gestion des interruptions non-masquables
 - Mode 0 : similaire au 8080. Le périphérique qui interrompt le CPU fournit également une instruction à exécuter. Mode par défaut après un RESET.
 - Mode 1 : saute en 0038.
 - Mode 2 : forme un ptr avec le registre I et les 8 bits fournis par le périphérique à l'origine de l'interruption.
 - → extrêmement souple

(ロ) (個) (E) (E) (E) (A

Introduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion

Jeux d'instructions

- 158 instructions dont 78 provenant du 8080.
 - Load and Exchange
 - Block Transfer and Search
 - Arithmetic and Logical
 - Rotate and Shift
 - Bit Manipulation (Set, Reset, Test)
 - Jump, Call, and Return
 - Input/Output
 - Basic CPU Control
- Entre 4 et 21 cycles.
- 1, 2, 3 ou 4 bytes.
- Pas de multiplication !



Introduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion **Pipelining**

- La séparation du bus de données et d'adresses permet l'overlapping.
- La plupart des opcodes sont codées sur un byte → Utilisation efficace de l'*overlapping* et minimise l'effet des bulles
- Le pipeline est très élémentaire. Seulement 2 niveaux : Fetch et Execute

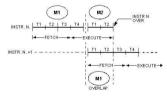


Figure: Fetch/Execute overlap



Introduction Présentation Registres Bus Interruptions Instructions Pipelining Conclusion Conclusion

- Présente de nombreux avantages p/r au 8080 tout en restant compatible avec celui-ci.
- Extrêmement populaire et toujours présent aujourd'hui.
- Très simple à déployer : CPU, oscilliateur, RAM, alim, I/O
- Relativement primitif: pas de gestion de cache, pas de MMU, pipelining très élémentaire, ...

