

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Codage et opérations binaires</b>	<b>3</b>
2.1	Notes . . . . .	3
2.2	Représentation des entiers non signés . . . . .	3
2.3	Représentation binaire des entiers signés . . . . .	4
2.4	Opérations arithmétiques sur les entiers signés . . . . .	6
2.5	Représentation des nombres réels . . . . .	6
2.6	Représentation des caractères . . . . .	6
<b>3</b>	<b>Eléments de circuits logiques</b>	<b>6</b>
3.1	Logique combinatoire . . . . .	6
3.1.1	Portes logiques élémentaires . . . . .	6
3.1.2	Circuits logique combinatoire . . . . .	6
3.2	Logique séquentielle . . . . .	6
3.2.1	Bascule RS . . . . .	6
3.2.2	Bascule D . . . . .	6
3.2.3	Bascule JK . . . . .	6
3.3	Synthèse . . . . .	7
<b>4</b>	<b>Instruction set architecture</b>	<b>7</b>
4.1	Instructions . . . . .	7
4.2	Décodeur . . . . .	7
4.3	Séquenceur . . . . .	7
<b>5</b>	<b>TD1/2 : utiliser logisim et séquencement à la main</b>	<b>7</b>
<b>6</b>	<b>TL1 : Câbler les éléments d'un microprocesseur</b>	<b>7</b>
<b>7</b>	<b>Compilateur, interpréteur</b>	<b>8</b>
7.1	langage bas niveau : microcode, assembleur . . . . .	8
7.2	langage haut niveau : python, c, ... interprété ou compilé . . . . .	8
7.3	chargement d'un programme en mémoire . . . . .	8
7.4	routines , pile/tas, stack pointer, .. . . .	8
<b>8</b>	<b>TD3/4 : a voir : transformer un code évolué en micro-code, ajouter pile avec appel de sous-routines.</b>	<b>8</b>
<b>9</b>	<b>Les différentes formes de mémoires</b>	<b>8</b>
9.1	Mémoire cache, RAM, ROM, disques durs (magnétiques, SSD, ..) . . . . .	8
9.2	systèmes de pagination ? . . . . .	8
<b>10</b>	<b>Interaction avec les périphériques</b>	<b>8</b>
10.1	Interruptions, conflits, .. . . .	8
<b>11</b>	<b>TD5/6 : Jouer avec la pyboard</b>	<b>8</b>
<b>12</b>	<b>TL : Jouer avec les interruptions, pyboard et afficheurs</b>	<b>8</b>
<b>13</b>	<b>Plus loin</b>	<b>8</b>
13.1	Pipelining . . . . .	8
13.2	Processeurs vectoriels, GPU, multi-coeurs, .. . . .	8