Automachef

MANUEL DE L'ORDINATEUR PROGRAMMABLE **AC-16**

TABLE DES MATIÈRES

ARCHITECTURE DU SYSTÈME

L'AC-16 est un ordinateur programmable de pointe capable de contrôler quatre appareils connectés sur les prises 00 à 03. Chaque sortie est déconnectée par défaut quand l'ordinateur est allumé et réinitialisé. Il est équipé de quatre registres 16 bits polyvalents, de V0 à V3. Ils sont aussi appelés variables dans ce manuel et peuvent stocker des entiers allant de -32 768 à 32 767.

Certaines pièces de la cuisine vous aideront à lire le nombre d'actions effectuées, comme les portes d'ingrédients, les assembleuses et les bras robotiques. Le nombre d'actions réalisées depuis la mise en service est accessible sur les variables I0 à I3. I0 lit les actions effectuées sur l'appareil connecté à 00, I1 et 01 etc. Les pièces qui ne lisent pas cette valeur (comme les tapis roulants) indiqueront 0.

La mémoire de l'ordinateur peut contenir 32 lignes de code écrites en Langage d'Assembleuse AC. Le programme est exécuté exactement 30 fois par seconde, pour faciliter les routines nécessitant un timing précis. D'ailleurs, en accédant au registre TT de lecture, vous trouverez l'heure au format 24h. À 3h45 de l'après-midi, le registre indiquera l'entier 1545 en 16 bits.

Vous disposez de quatre registres spéciaux qui recevront automatiquement les données de la trieuse intégrée. Chacun d'eux indiquera un chiffre supérieur à 0 si au moins une nouvelle commande du type spécifié est arrivée dans les 33 millisecondes. Cette valeur détermine combien de ces commandes ont été reçues dans les 33 millisecondes. La référence se trouve dans les instructions R0 à R3.

Les trieuses intégrées peuvent aussi reconnaître les commandes venant d'autres sources, comme le Drive-in ou les clients souhaitant un plat à emporter. Vous pouvez ainsi ajouter les lettres suivantes aux variables R0 à R3 :

R pour les commandes venant du restaurant.

T pour les commandes de la zone à emporter.

D pour les commandes du Drive-in.

Exemples

Trieuse intégrée R2 pour détecter les Cheeseburgers.

La variable R2R indiquera le nombre de Cheeseburgers commandés depuis le restaurant.

La variable R2T indiquera le nombre de Cheeseburgers commandés depuis la zone à emporter.

La variable R2D indiquera le nombre de Cheeseburgers commandés depuis le Drive-in.

La variable R2 indiquera le nombre total de Cheeseburgers commandés depuis 33 millisecondes.

R2 fera donc la somme de R2R + R2T + R2D.

LEXIQUE

ÉTIQUETTES

Les étiquettes sont des emplacements du code que vous pouvez utiliser pour changer le flux d'exécutions avec les commandes jump (jmp/jne/jeq/jgt/jlt). Elles peuvent contenir des caractères alphabétiques, au format 1 à 10 caractères, suivi de deux points.

EXEMPLES

```
loopagain:
```

```
belton:
```

```
endprogram:
```

COMMANDE ADD

La commande Ajouter insère deux valeurs et stocke le résultat de la variable spécifiée dans le troisième paramètre. N'oubliez pas, les registres sont en 16 bits, le résultat doit donc se trouver entre -32 768 et 32 767 pour éviter toute erreur.

Syntaxe

```
add <operand1> <operand2> <operand3>
<operand1> peut être une variable ou un entier.
<operand2> peut être une variable ou un entier.
<operand3> doit être une variable.
```

Exemples

```
add V1 15 V2
```

```
add V0 V1 V0
```

COMMANDE CMP

La commande cmp compare deux valeurs et paramètre le registre de comparaison sur -1, 0 ou 1. Si la première valeur est plus petite que la seconde, elle indiquera -1. Si elles sont égales, elle indiquera 0. Si la première valeur est plus grande, elle indiquera 1. Le résultat peut servir à sauter à une autre partie du code avec les commandes jne/jeq/jlt/jgt.

Syntaxe

```
cmp <operand1> <operand2>
<operand1> peut être une variable ou un entier.
<operand2> peut être une variable ou un entier.
```

Exemples

```
cmp V1 30
```

```
cmp V1 V3
```

COMMANDE DEC

La commande dec réduit la variable spécifiée de 1, mais ne permet jamais de tomber à des valeurs négatives, puisqu'elle s'arrêtera à 0. Très pratique pour intégrer les chronomètres.

SYNTAXE

```
dec <operand1>
<operand1> doit être une variable.
```

EXEMPLES

```
dec V0
```

```
dec V3
```

COMMANDES JMP/JNE/JEQ/JGT/JLT

Toutes ces commandes sautent vers l'étiquette spécifiée. jne signifie "sauter si pas égal", jeq "sauter si égal", jlt "sauter si inférieur à" et jgt "sauter si supérieur à". Elles sauteront à l'étiquette spécifiée si le résultat de la dernière comparaison à effectuer avec la commande cmp correspond à celui de la commande. Exemple : lancer la commande jne après une comparaison fera sauter sur l'étiquette uniquement si cette comparaison indique que les paramètres n'étaient pas égaux. Une

commande jgt ne sautera que si le premier paramètre était supérieur au second etc. La commande jmp sautera toujours (sans condition) vers l'étiquette spécifiée.

Syntaxe

```
jmp <operand1>
jne <operand1>
jeq <operand1>
jlt <operand1>
jgt <operand1>
<operand1> doit être une étiquette.
```

Exemples

```
jne endprogram
```

```
jlt loopagain
```

COMMANDE MOV

La commande mov copie une valeur dans la variable spécifiée.

Syntaxe

```
mov <operand1> <operand2>
<operand1> peut être une variable ou un entier.
<operand2> doit être une variable.
```

Exemples

```
mov 30 V2
```

```
mov V1 V2
```

COMMANDE NOT

Cette commande se contente d'inverser la valeur d'une variable. De 0, elle passe à 1 et inversement.

Syntaxe

```
not <operand1>
<operand1> doit être une variable.
```

Exemples

```
not V1
```

```
not V3
```

COMMANDE OUT

Cette commande allume ou éteint l'appareil branché à la sortie indiquée. Si le deuxième opérande est 0, il sera éteint. Toute autre valeur activera la sortie.

Syntaxe

```
out <operand1> <operand2>
<operand1> doit être une sortie.
<operand2> peut être une variable ou un entier. 0 signifie "éteint",
les autres valeurs "allumé".
```

Exemples

```
out O2 1
```

```
out O2 V3
```

COMMANDE RET

La commande ret (retour) termine l'exécution du code sur un cycle de 33 millisecondes. Elle est identique à un saut vers une étiquette placée sur la dernière ligne de code. Aucun opérande.

Syntaxe

```
ret
```

Exemples

```
ret
```

COMMANDE SUB

La commande sub calcule la différence entre deux valeurs et stocke le résultat de la variable spécifiée dans le troisième paramètre. N'oubliez pas, les registres sont en 16 bits, le résultat doit donc se trouver entre -32 768 et 32 767 pour éviter toute erreur. En bref : elle calcule operand1 - operand2 et stocke le résultat dans la variable spécifiée dans operand3.

SYNTAXE

```
sub <operand1> <operand2> <operand3>
<operand1> can be a variable or an integer value.
<operand2> can be a variable or an integer value.
<operand3> has to be a variable.
```

EXEMPLES

```
sub V1 15 V1
```

```
sub V2 V3 V0
```

EXEMPLE DE PROGRAMME 1

Ce programme allume l'appareil connecté à O1 pendant une seconde, l'éteint pendant 1 seconde etc.

```
add 1 V0 V0
cmp 30 V0
jne endprogram
mov 0 V0
not V1
out O1 V1

endprogram:
```

EXEMPLE DE PROGRAMME 2

Ce programme allume l'appareil connecté à O2 après 5 commandes reçues.

```
add V0 R0 V0
cmp V0 5
jlt endprogram
out O2 1

endprogram:
```

EXEMPLE DE PROGRAMME 3

Ce programme garde l'appareil connecté à O0 allumé pendant 5

secondes à chaque nouvelle commande reçue. Il sera dès lors très utile pour indiquer au Distributeur de produire un ingrédient pour chaque commande arrivée. Remarque : il préchauffera aussi l'appareil 4 secondes à l'allumage, de sorte que les ingrédients sont servis presque immédiatement après l'arrivée de la commande... un gain de temps précieux pour les clients ! Une Trieuse classique en serait incapable !

```
prewarm:
cmp V1 1
jeq alrdywarm
add 120 V0 V0
mov 1 V1

alrdywarm:
cmp R0 1
jne nonew
add 150 V0 V0

nonew:
cmp V0 0
jlt timerended
sub V0 1 V0
out O0 1
ret

timerended:
out O0 0
```

EXEMPLE DE PROGRAMME 4

Ce programme plus complexe lit deux modules de tri de commande différents (R0 et R1) et gère les sorties O0, O1 et O2. L'objectif est d'allumer O0 et O1 trois secondes, chaque fois qu'une nouvelle commande arrive sur R0 ou R1, mais de n'allumer O2 que si une nouvelle commande se présente sur R1. Très utile pour indiquer à R0

de ne traiter que les burgers classiques et à R1 les cheeseburgers. O0 et connecté au Distributeur de steak haché cru, O1 à un Distributeur de pain burger et O2 à un Distributeur de fromage. Il préchauffe aussi pendant 2 secondes à l'allumage.

```
prewarm:
cmp V2 1
jeq checkorder
add V0 60 V0
add V1 60 V1
mov 1 V2

checkorder:
cmp R0 1
jeq addtime
cmp R1 1
jeq addtimes

main:
out O0 V0
out O1 V0
out O2 V1
dec V0
dec V1
ret

addtimes:
add V1 90 V1
addtime:
add V0 90 V0
jmp main
```