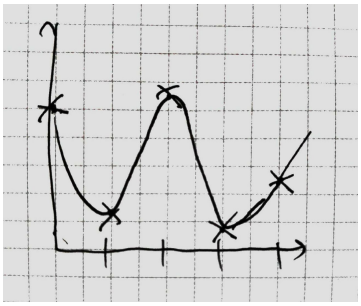
Klausur WS16/17

Aufgabe 1

Teilaufgabe 1

a) Regelung ist eine Steuerung mit zusätzlich einem Feedback Loop (also Output-Wert wird verwendet um Steuerung zu korrigieren)
b) https://en.wikipedia.org/wiki/Closed-loop_transfer_function
$G_{oben} = G_1(G_2/(1 + G_2G_3))G_4 = G_1G_2G_4/(1 + G_2G_3)$ $G = G_{oben}/(1 - G_{oben})$
c) X(s)*(2s^2-7s)=-2W(s) => G(s)=X(s)/W(s) = -s^2+3.5s
d) Nullstellen des Nenners: $s_{12} = (-2 \pm \sqrt{4 - 12})/2 = -1 \pm \sqrt{2}j$ -> Realteil aller Nullstellen negativ -> stabil
e) Nein, das ist das D-Glied

Teilaufgabe 2

a) 1/40kHz=1/40 ms = 25us (Shannon-Nyquist Theorem, max frequenz *kleiner* dh abtastrate 2x reicht)
b) (x(k)-x(k-1))/T - x(k-1)-x(k-2))/T)/T - 2*x(k)=(w(k)-w(k-1)/T + w(k)
c) Im hinweis sollte f_k -> f_i sein? $x_k(3/z^2 + 1/z + 2) = w_k + z(w_k - w_0z) + z^2(w_k - w_0 - w_1/z)$
w_0=0,w_1=0
$G(z) = x_k/w_k = (1 + z + z^2)/(3/z^2 + 1/z + 2)$
d) Nullstellen: (1, 0), (-2, 0); Polstellen: (-2, 0), (0, sqrt(½)), (0, -sqrt(½))
e) Nicht stabil, da Betrag der ersten Polstelle = 2
f) 

Aufgabe 2

Teilaufgabe 1

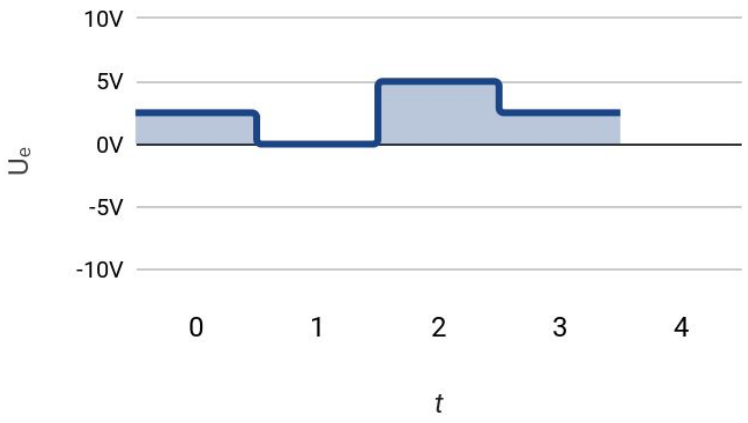
a) Überwacht ein Programm, wenn der Zähler abläuft wird das Programm getötet, weil davon ausgegangen wird dass es hängen geblieben ist. Zum verhindern sendet das Programm regelmäßig ein Signal an den Watchdog.
b) Das Ausführen von Befehlen besteht aus mehreren Schritten, während die späteren Schritte (zb Execute) des ersten Befehls laufen können schon die früheren Schritte des nächsten Befehls ausgeführt werden (zb Load)
c) Kennzahlen: zunehmende Größe, abnehmende Geschwindigkeit.
Register-Cache-Hauptspeicher-Festplatte
d) PCI-Bridge. Komponenten auf verschiedenen Bussen können miteinander verknüpft werden (zb PCIe-to-PCI-Bridge)
e) 8b/10b encoding, 1-nutzdaten=1-8/10=20% overhead (??)
f) 4x0.5=2GByte/s (??)

Teilaufgabe 2

a) Ein OP in Grundform ist ein unendlicher Differenzverstärker (?). Rin=inf, Rout=0, V=inf

b) Invertierender Addierer **(gleiche aufgabe wie SS2018:Aufgabe2.2)**
Maschengleichungen, mithilfe virtueller Erde zw - und + und +=Ground:
$U_a = R_2 I_2$
$U_e = R_{11} I_{11} \Rightarrow I_{11} = U_e / R_{11}$
$5V = R_{12} I_{12} \Rightarrow I_{12} = 5V / R_{12}$
Knotengleichung:
$I_2 = -(I_{12} + I_{11})$
ergibt:
$U_a = R_2 I_2 = -R_2 (U_e / R_{11} + 5V / R_{12})$

Zur skalierung zw 0 und 5V muss ein Wert -> 0 und einer -> 5V. Da die Schaltung invertierend ist, wird der kleinere Wert zum größeren (-0.4V→5V) und der größere zum minimum -0.2V→0V
(allgemein ist die formel zur Normierung $normvalue = (value - min)/(max - min)$
und zur Skalierung $value = normvalue * (maxnew - minnew) + minnew$)
dh das Ergebnis muss sein $U_a = -(5/(0.4 - 0.2) * (U_e - (0 - 0.2V)) = -25(U_e + 0.2V)$
d.h. z.B. $R_2 = 25\ Ohm$, $R_{11} = 1\ Ohm$, $R_{12} = 5V/0.2V = 25\ Ohm$



c) Parallelverfahren, für jede mögliche Stufe ein Vergleicher
U_LSB=6V/4bit = 1.5V. Dh Stufen sind 1.5V, 3V, 4.5V.
Hauptvorteil: Instant, alle anderen brauchen mehrere Schritte

Aufgabe 3

Teilaufgabe 1

a) CSMA/CA, CSMA/CD, Token Passing, Polling, TDMA
b) Glasfaser, Twisted Pair, (?)
c) 01100101101
d) I6
e) 4/2/3/5/6/-/2

Teilaufgabe 2

a) logische Korrektheit, zeitliche Korrektheit
b) ?? periodische und aperiodische zeitbedingungen?
c) Asynchrone Programmierung

d) Statische/dynamische Priorität, Präemptiv/Kooperativ
e) Ein Scheduler der wenn ein gültiger Schedule existiert ihn auch findet
f) Nein: zb Wenn T2 max Priorität hat, und T1 und T2 gleichzeitig ankommen, wird T1 verpasst.
g) 30%+25%+20% = 75%. Wenn Prozessorauslastung > n*(2^(1/n)-1) = 3*(2^(⅓)-1) = 3*0.26=78%. findet es sicher keinen. nur wenn RMS verwendet wird findet es sicher einen mit dieser auslastung

Aufgabe 4

Teilaufgabe 1

a) Echtzeit-IPC (mit end-zu-end-prioritäten), Ressourcenverwaltung (Synchronisation, Priority Inheritance), Echtzeitfähige Speicherverwaltung (keine Verdrängung)
b) Deadlock=Programme warten für immer auf gegenseitig blockierte Ressource. Starvation=hochpriorer Task verhindert längerfristig das niedrigpriorier laufen kann.
Inversion=Task A, B, C mit Prioritäten A>C>B. B blockiert eine Ressource. A will auch die Ressource, muss aber warten. Jetzt Kommt C an und starvt B und damit auch A. Effektiv hat C also höhere Priorität als A.
c) Lineare: Adressraum von Prozess ist vollständig direkt in den physischen Speicher abgebildet mit einem Offset. Streuend: Seiten aus dem virtuellen Addressraum werden per Page Table in Kacheln im physischen Speicher abgebildet. Die Reihenfolge der Kacheln ist unabhängig von der Reihenfolge der Seiten.
d) Streuend ist besser: Wenn bei linearer Adressbildung Fragmentierung auftritt, müssen beim erstellen eines neuen Segments andere Prozesse pausiert werden um sie zu verschieben. Bei Paging können einfach die freien Kacheln wieder benutzt werden. Und Dynamische Taskgrößenänderungen sind einfacher und zeitlich besser vorhersagbar. Nachteil von Paging ist der höhere Rechenaufwand und der unnutzbare Speicher wenn ein Prozess nur einen Teil einer Seite benötigt.

Teilaufgabe 2

SPS ist nicht mehr teil der vl